МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
“ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ”

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
«Основы программирования в среде Visual Basic»
из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Харьков НТУ «ХПИ» 2013

Методические указания к лабораторным работам «Основы программирования на Visual Basic» из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки» дисциплины «Информатика» для студентов направления подготовки 6.050801 «Микро- и наноэлектроника» / Состав.: В.И. Шкалето, Р.В. Зайцев. – Харьков: НТУ «ХПИ», 2013. – 60 с.

Составители: В.И. Шкалето,
Р.В. Зайцев,
М.В. Кириченко

Рецензент доц. Г.И. Копач

Кафедра физического материаловедения для электроники и гелиоэнергетики

ВВЕДЕНИЕ

Методические указания к лабораторным раотам по разделу «Программирование в среде Visual Basic. Интегрированная среда разработки» дисциплины «Информатика» касаются четырех лабораторных работ: «Файлы. Дескриптор файла. Типы доступа. Ввод и вывод данных на диск.», «Типы ошибок. Инструменты отладки проектов. Работа с трехмерной графикой.», «Построение графиков функций методами Visual Basic. Использование MSFlexGreed.» и «Моделирование физических зависимостей с использованием Visual Basic».

Так что же такое Visual Basic? Слово "Visual" относится к методу, используемому для создания того, что видит пользователь — графического пользовательского интерфейса, или GUI (graphical user interface). Слово "Basic" относится к аббревиатуре BASIC (Beginners All-Purpose Symbolic Instruction Code — многоцелевой код символьных инструкций для начинающих) языка программирования, который используется программистами намного чаще, чем любой другой язык в истории вычислений. Для создания различных полезных программ достаточно изучить лишь некоторые из его возможностей. Приведенные ниже лабораторные работы содержат материалы для начинающих программистов в среде Visual Basic 6.0 (VB6.0), в каждой из которых изучаются определенные вопросы технологии работы со средой программирования VB6.0 и решаются различные варианты учебной задачи.

В результате изучения теоретического материала и выполнения лабораторных работ студент должен изучить:

- этапы разработки программ;
- основы алгоритмического языка VB6.0;
- назначение, функции и структуру среды программирования VB6.0;

а также уметь:

- проводить физический и математический анализ простейших прикладных задач и разрабатывать алгоритмы их решения;
- вести отладку программ в среде программирования VB6.0.

Указанное позволит преорести первоначальные навыки работы в среде программирования VB6.0

ЛАБОРАТОРНАЯ РАБОТА 1

ФАЙЛЫ. ДЕСКРИПТОР ФАЙЛА. ТИПЫ ДОСТУПА. ВВОД И ВЫВОД ДАННЫХ НА ДИСК.

Цель работы – Научиться работать с файлами. Создать проект на Visual Basic 6.0, содержащий формы, различные элементы управления, процедуры и функции. Научиться выводить информацию в файл и вводить ее из файла. Научиться обрабатывать графические, текстовые и математические данные.

1.1 Общие сведения

1.1.1 Ввод и вывод информации

Рассмотрим различные возможности ввода и вывода информации в Visual Basic, в частности, ввод и вывод информации на различные носители данных (жесткий диск, дискета), а также на принтер.

Информацию зачастую требуется не только анализировать, но и сохранять. Для сохранения информации предназначены операторы обработки файлов, позволяющие считывать и сохранять данные на различных носителях (гибкий либо жесткий диск и т.п.). Процесс открытия и сохранения файлов состоит из нескольких этапов:

- получение дескриптора файла (handle);
- открытие файла;
- чтение или запись данных;
- закрытие файла.

Чтобы работать с файлами, нужно понимать, как связывается система или приложение с файлом. Для этого имеется канал ввода/вывода. При открытии файлу ставится в соответствие канал с определенным номером. Таким образом, каждый открытый файл имеет собственный канал, с помощью которого записываются или считываются данные. Следовательно, для ввода и вывода данных в файл имеет значение не имя файла, а номер канала. Кроме того, операционная система должна иметь сведения о наличии свободных каналов, которые можно использовать для открытия файла.

```
FreeFile
```

Функция Visual Basic FreeFile возвращает номер свободного канала, который можно использовать для работы с файлом.

```
FreeFile [(RangeNumber)]
```

Если свободных каналов нет (открыто максимально допустимое количество файлов), возникает ошибка вы-полнения.

```
intFH = FreeFile ()
```

В этом примере переменной intFH присваивается целое значение, которое можно использовать для открытия файла. Необязательный параметр RangeNumber позволяет определить диапазон значений, из которого

выбирается очередной свободный номер канала. Если его значение равно 0 (по умолчанию), то возвращается номер канала из диапазона 1 — 255, если 1, то из диапазона 256 — 511.

1.1.2 Типы доступа

В Visual Basic реализованы три типа доступа к файлам:

• последовательный (Sequential) — для чтения и записи текстовых файлов;

• произвольный (Random) — для чтения и записи текста или структурированных двоичных файлов с записями фиксированной длины;

• двоичный (Binary)— для чтения и записи произвольно структурированных файлов.

При создании коммуникационных каналов система должна знать, какой тип доступа к каждому конкретному файлу нужно использовать и какова структура данных этого файла.

1.1.2.1. Последовательный доступ

Последовательный доступ используется главным образом при работе с текстовыми файлами. Любая информация считывается или сохраняется в текстовом виде построчно. В тексте могут находиться символ перевода строки (vbCrLf или Chr(13)&Chr(10)) или табулятор (vbTab или Chr(9)). Эти символы используются для форматирования текста.

Способ открытия файла с последовательным доступом (для чтения, записи или добавления) задается при вызове оператора Open:

```
Open Имя_файла For [Input | Output | Append ] As #Filehandle
```

Таблица 1.1. Различные операционные возможности для последовательного доступа

Ключевое слово	Описание
Input	Открытие только для чтения из файла
Output	Открытие для записи в файл
Append	Открытие для добавления к файлу

Если файл не существует и открывается для чтения (For Input), то Visual Basic выдает сообщение об ошибке, а если для записи или добавления (Output или Append), то создается новый файл. Если файл с указанным именем существует, то в режиме Output его содержимое удаляется, а в режим Append файл открывается для добавления:

```
Open "C:\README.TXT" For Input As #intFH1
Open "C:\DATA\TEXT.TXT" For Output As #intFH2
Open "C:\USERS.TXT" For Append As #intFH3
```

В конце строки указывается номер канала, возвращаемый функцией FreeFile. В некоторых операционных системах, например в Windows, можно использовать длинные имена файлов.

Для считывания данных из файла, открытого для последовательного доступа, существует несколько возможностей. В общем случае это осуществляется с помощью оператора Input, имеющего несколько разновидностей:

• Line Input # считывает одну строку;

• Input #считывает последовательность символов, обычно записанных с помощью оператора Write #;

• Input$ #считывает определенное количество символов.

Существует несколько вариантов чтения всей информации из файла. Перед чтением нужно открыть файл с помощью оператора Open...For:

```
intFH = FreeFile
Open "C:\Text.Txt"For Input As intFH
' 1-ый вариант
Do Until EOF(intFH)
Line Input #intFH, strString
strText =strText & strString & vbLf
Loop
Close #intFH
' 2-ой вариант
strText = Input$(LOF(intFH), intFH)
Close #intFH
```

Оба варианта приводят к одинаковому результату. В первом варианте оператор Input выполняется в цикле, пока не будет достигнут конец файла. Функция EOF (End Of File) возвращает значение True при достижении конца файла. При этом на каждом шаге цикла считывается отдельная строка и к ней добавляется символ конца строки, который отбрасывается оператором Line Input. Во втором варианте весь файл считывается функцией Input$. Функция LOF (Length Of File) позволяет определить длину файла в байтах. Заметим также, что независимо от вида оператора Input указывается не Имя файла, а только номер канала, т.е. дескриптор файла (intFH).

Оператор Close предназначен для закрытия открытого файла или канала.

В Visual Basic для записи информации в файл используются операторы Print # и Write #.

Оператор Print # функционирует почти так же, как его "коллега" для экрана, с той лишь разницей, что данные не выводятся на экран, а сохраняются в файле, открытом для записи или добавления (Open…For Output или Open…For Append).

```
Print #filehandle, [Spc(n) | Tab [(n)] |] [expression]
[charpos]
```

Синтаксис оператора на первый взгляд выглядит сложно:

```
Print #intFH, Text1.Text
Print #intFH, "Фрагмент1", "Фрагмент2"
Print #intFH, "Это составляет "; "единое целое"
```

Для форматирования записываемой в файл информации следует поразному разделять данные в операторе Print #. Если в операторе данные разделять запятыми (,), то в файле они будут разделены символами табуляции:

```
Print #intFH, "Фрагмент1", "Фрагмент2"
'соответствует
Print #intFH, "Фрагмент1"; Tab; "Фрагмент2"
```

Если же в операторе для разделения данных использовать точку с запятой (;), то данные в файл записывают-ся без разделителей:

```
Print #intFH, "Это составляет "; "единое целое"
'соответствует
Print #intFH, "Это составляет единое целое"
```

Оператор Write # имеет такой же синтаксис, что и Print #. Отличие состоит только в форматировании вывода. Если Print # сохраняет данные в виде обычного текста, то Write # заключает текстовые строки в кавычки, а цифры выводятся без кавычек:

```
Print #intFH, "Анна", "Киев", 17
' в файле будет: Анна
Write #intFH, "Анна", "Киев", 17
'в файле будет: "Анна", "Киев", 17
```

Данные, сохраненные с помощью оператора Write #, можно считать оператором Input #.

1.1.2.2. Произвольный доступ

В отличие от последовательного доступа, при котором данные в файлах хранятся в неструктурированном виде, произвольный доступ предполагает, что файл имеет постоянную структуру. Это позволяет считывать данные в произвольном порядке.

Произвольный доступ реализуется посредством оператора Open.

```
Open Имя_файла For Random [Access] [Lock] As [#]Handle
[Len = Длина_записи]
```

Параметр Len определяет длину записи. Если это значение меньше, чем реальная длина записи, то возникает ошибка, если больше — то при записи файла используется больше дискового пространства, чем необходимо.

Параметр Access позволяет задать права доступа к открываемому файлу.

Таблица 1.2. Виды доступа при произвольном доступе

Доступ	Пример
Без указания	Open “DATE.DAT” For Random As intFH
Чтение (Read)	Open “DATE.DAT” For Random Access Read As intFH
Запись(Write)	Open “DATE.DAT” For Random Access Write As intFH
Чтение и запись (Read Write)	Open “DATE.DAT” For Random Access Read Write As intFH

Если права доступа не указаны, то по умолчанию используется Read Write. Так как этот тип доступа обычно предназначен для работы с файлами, которые могут использоваться многими пользователями или приложениями, то следует обеспечить целостность данных при коллективном использовании. Для этого следует установить параметр Lock, определяющий права доступа к открытому файлу.

```
Open “DATE.DAT” For Random Access Read Write As intFH
```

Этот параметр может принимать следующие значения:

• Shared - Файл может использоваться всеми процессами для считывания и записи.

• Lock Read - Никакой другой процесс не может считывать данные из файла. Данный параметр можно установить, если в данный момент никакой другой процесс не выполняет операцию чтения.

• Lock Write - Никакой другой процесс не может записывать данные в файл. Данный параметр можно установить, если в данный момент никакой другой процесс не выполняет операцию записи.

• Lock Read Write - Никакой другой процесс не может считывать или записывать. Данный параметр можно установить, если в данный момент не выполняются операции чтения или записи.

Параметр Len задает длину одной записи.

Для записи и чтения данных используются соответственно операторы Put и Get.

```
Put #filehandle, Номер_записи, Переменная
Get #filehandle, Номер_записи, Переменная
```

В данном примере в файл записываются данные из переменной Address, причем номер записи равен 7, а затем в переменную Address считывается вторая запись файла.

```
Put #intFH, 7, Address 'сохраняет 7-ую запись
Get #intFH, 2, Address 'считывает 2-ую запись
```

Для того чтобы в одной записи сохранить несколько значений различных типов, следует использовать пользовательские типы данных:

1.1.2.3. Двоичный доступ

Для открытия двоичного файла также используется оператор Open.

```
Open Имя_файла For Binary [Access] [Lock] As [#]Handle
```

Формат оператора Open при двоичном доступе похож на формат этого оператора при произвольном доступе. Главное отличие состоит в том, что вместо ключевого слова Random указывается Binary, а параметр Len отсутствует, так как записи имеют фиксированную длину — 1 байт

```
Open "SPOCK VUL" For Binary As intFH
```

Для считывания и записи в файл используются также операторы Get и Put Синтаксис их такой же, как при произвольном доступе, только вместо номера записи указывается номер байта.

```
Get #intPH, 5, Var
Put # intPH, 78, Var
```

Если данные считываются в строку, количество считываемых байтов равно длине строки:

```
strIutput$ = String (14, " )
Get #intFH, 26, str0utput$
```

В данном примере в строку считывается 14 байтов, начиная с 26-го байта файла

Для корректного завершения работы приложения все открытые файлы следует закрыть. Закрытие файлов выполняет оператор Close:

```
Close #intFH
Close intFH
Close
```

Если дескриптор указан, то закрывается соответствующий файл, если нет, то закрываются все открытые файлы.

1.1.3. Шрифты

1.1.3.1. Технологии шрифтов Windows

Различия в шрифтах вызывают проблему отображения данных на экране, если, например, в документе применяется принтерный шрифт, не являющийся экранным. В Windows используются шрифты следующих типов:

- растровые;
- векторные;
- TrueType (контурные);
- принтерные.

Шрифты Windows можно подразделить также на масштабируемые и не масштабируемые. Размер масштабируемого шрифта может изменяться произвольно. Для не масштабируемых шрифтов размер символов может принимать только строго фиксированные значения, заданные в файле шрифта.

Для растрового шрифта все символы хранятся в файле в виде растровых изображений и выводятся на экран или принтер как массивы точек. Растровые шрифты не являются масштабируемыми. В окне настройки шрифтов (Панель управления \Шрифты) растровые шрифты легко узнать по пиктограмме в виде красной буквы А с указанными в названии размерами.

Векторные шрифты создаются путем соединения заданных точек прямой линией. В связи с тем, что расстояние между точками может легко изменяться, векторные шрифты являются масштабируемыми.

В окне настройки шрифтов (Панель управления \Шрифты) векторные шрифты также отображаются пиктограммой в виде красной буквы А, но без указания размера. (Размер может не указываться и для некоторых растровых шрифтов.)

Для шрифтов типа TrueType (контурный) заданные точки соединяются прямыми или кривыми линиями, формируя контурное очертание

символа. Изображение символа получается путем закрашивания этого контура. Поэтому шрифтовые файлы относительно малы. Шрифты этого типа являются свободно масштабируемыми.

Преимущество этой технологии в том, что созданные шрифты могут выводиться как на экран, так и на принтер, при этом соблюдается принцип WYSIWYG ("What you see is what you get"— ""Что видите, то и получаете"). Поскольку эта технология известна под названием "TrueType", на пиктограмме шрифтов этого типа изображены две буквы T — серая и голубая.

В отличие от шрифтов других видов, принтерные шрифты хранятся не в файле, а встроены в устройство печати. Принтерные шрифты не видны в окне настройки шрифтов (Панель управления \Шрифты). Их можно увидеть только в диалоговом окне выбора шрифта, где рядом с именем шрифта расположена пиктограмма с изображе-нием принтера.

1.1.3.2. Объект Font

Для выбора и установки параметров шрифта (наименование, размер, дополнительные атрибуты) в Visual Basic существует объект Font. Этот объект определяет вид шрифта другого объекта (TextBox, Form, Printer и т.д.).

Объект Font имеет ряд свойств, с помощью которых выполняются нужные установки.

Свойство Name объекта Font содержит имя шрифта. Чтобы просмотреть список всех шрифтов, доступных для использования, можно воспользоваться свойством Fonts объекта Printer или Screen (принтерные или экранные шрифты):

```
For i =0 To Screen.FontCount - 1
lstFont.AddItem Screen.Fonts(i)
Next i
```

Свойство FontCount возвращает количество доступных шрифтов.

Свойство Size указывает размер шрифта в пунктах и может принимать значения в диапазоне от 1 до 2160.

Свойства Bold, Italic, Underline и StrikeThrough позволяют установить атрибуты шрифта: полужирный, курсив, подчеркивание и перечеркивание. Возможными значениями для этих свойств являются True или False.

Значение свойства Weight, задаваемое пользователем или разработчиком, округляется до одного из двух фиксированных значений: 400 (обычная толщина) или 700 (полужирный).

Некоторые элементы управления и объекты Visual Basic (например, Form, PictureBox, Printer) обладают специфическим свойством, связанным с отображением шрифтов – FontTransparent. Если его значение равно False, то при выводе текста фон под символами не отображается; в противном случае символы отображаются поверх существующего фона.

Функцию, аналогичную FontTransparent, для некоторых элементов управления (например, Label) выполняет свойство BackStyle. Если его значение равно 0, то фон элемента управления будет прозрачным.

Кроме объекта Font, в Visual Basic есть и свойство Font. Это свойство содержит ссылку на объект Font и может использоваться так же, как и объект Font.

1.1.3.3. Диалоговое окно изменения вида шрифта

Файл COMDLG32.DLL содержит стандартные диалоговые окна Windows, одно из которых используется для установки параметров шрифта. Это диалоговое окно отображается с помощью метода ShowFont или свойства Action:

```
CommonDialog1.ShowFont
```

или

```
CommonDialog1.Action = 4
```

Однако такая попытка открытия окна (без предварительной настройки параметров) вызывает ошибку под номером 24574 — "No fonts exist"("Нет шрифтов"). Это связано с тем, что диалоговое окно должно "знать", какого вида шрифты — принтерные или экранные — следует отображать. Поэтому перед открытием диалогового окна нужно установить свойство Flags равным cdlCFScreenFonts, cdlCFPrinterFonts или cdlCFBoth, после чего диалоговое окно будет отображать шрифты указанного типа (типов). Если не задать значение этого свойства, то возникает ошибка выполнения.

```
CommonDialog1.Flags = cdlCFPrinterFonts
CommonDialog1.ShowFont
```

Выбор шрифта можно ограничить присвоением свойству Flags значений дополнительных констант. Например, константа cdlCFForceFontExist задает вызов сообщения об ошибке при попытке пользователя выбрать несуществующий шрифт или стиль, а константа cdlCFLimitSize указывает, что диалоговое окно будет отображать размеры шрифтов из диапазона, задаваемого свойствами Min и Max:

```
CommonDialog1.Flags =cdlCFPrinterFonts Or cdlCFForceFontExist Or cdlCFLimitSize
CommonDialog1.Min = 8
CommonDialog1.Max =20
CommonDialog1.ShowFont
```

Возможности диалогового окна можно при желании расширить. Константа cdlCFEffects позволяет отобразить дополнительные параметры форматирования для перечеркивания, подчеркивания и цвета.

Заданные пользователем значения параметров шрифта должны быть обработаны программой. Для этого приложение считывает значения свойств FontName, Font Size, FontBold, Fontltalic, FontStrikethru, FontUnderline и Color объекта CommonDialog. После установки всех необходимых параметров в диалоговом окне пользователь может завершить

его работу либо кнопкой ОК, т.е. принять все изменения, либо кнопкой Отмена, чтобы отказаться от них.

1.1.4 Печать

1.1.4.1. Метод PrintForm

С помощью метода PrintForm на принтер выводится форма в виде растрового изображения с установленным в системе разрешением (чаще всего 96 dpi).

Метод PrintForm может использоваться только для печати форм. При этом на принтер, установленный по умолчанию, выводится только содержимое формы, без строки заголовка и рамки. Все элементы управления выводятся на печать так, как отображаются на экране, т.е. с соответствующими надписями, границами, видами шрифтов и т.д. Преимущество метода PrintForm заключается в том, что форма выводится на принтер в том виде, в котором она отображается на экране, и для выполнения этого достаточно одной строки кода.

Недостаток метода, как это ни парадоксально, состоит в том, что в большинстве случаев форма в том виде, в котором она отображается на экране, чаще всего не нужна на бумаге. Еще одной проблемой является то, что формы могут иметь любой размер, поэтому не всегда возможно с помощью метода PrintForm полностью напечатать на лист формата А4. В зависимости от установленного разрешения экрана форма имеет соответствующие максимальные размеры.

1.1.4.2 Объект Printer

Объект Printer — это объект, предназначенный для вывода на печать текста и графики. В отличие от метода PrintForm, объект Printer позволяет выводить документ на печать с разрешением, установленным для принтера, а не для экрана, благодаря чему можно достичь лучшего качества печати.

Главным недостатком объекта Printer, вытекающим из его же достоинства, является необходимость программирования процесса печати, требующего написания объемного кода.

Наиболее важным методом объекта Printer является Print – передача текста на принтер:

```
Printer.Print "Hello"
Printer.Print "World"
```

Вывод осуществляется, начиная с верхнего левого угла печатной страницы, с использованием текущих параметров объекта Printer. Для изменения вида шрифта используются свойства объекта Font.

Для создания печатного документа нужно, кроме форматирования, установить также позицию точки вывода.

Объект Printer — это независимый от устройства объект (device independent). Это значит, что объект "не знает", на какое устройство вывода осуществляется печать. Поэтому для установки позиции должны испо-

льзоваться также независимые от устройства единицы измерения. С этой целью была введена единица измерения твип (twip). Не зависимо от модели принтера, 567 твипов составляют 1 сантиметр. Благодаря этому можно разрабатывать программы, качество печати которых не зависит от текущих драйверов печати.

Если единица измерения твип кажется вам непривычной, то с помощью свойства ScaleMode можно установить и другую. Константы vbCentimeters, vbMillimeters или vbPixels позволяют использовать знакомые вам единицы измерения.

Для изменения позиции точки вывода используются свойства CurrentX и CurrentY. Они подобны свойствам Top и Left, посредством которых задается расстояние от левого верхнего края печатаемой области до точки вывода. Ширина и высота печатаемой области в условных единицах измерения устанавливаются свойствами ScaleWidth и ScaleHeight.

При использовании пропорциональных шрифтов ширина слова зависит не только от количества букв, так как ширина букв может быть разная (например, W и i).В этом случае для правильного вывода текста следует определять фактическую ширину и высоту строки. Для этого используются методы TextHeight и TextWidth объекта Printer.

При выводе на печать для позиционирования текста могут использоваться табуляторы. Если в операторе Print выводимые значения разделяются точкой с запятой (;), то они печатаются один за другим, без разделителя, а если с запятой (,), — каждое очередное значение печатается в начале следующей зоны печати. Зоны печати в Visual Basic начинаются через каждые 14 символов (т.е. в столбце 0, столбце 14,столбце 28 и т.д.). Изменить ширину зоны позволяет функция Tab:

```
Printer.Print "Hello"; "World"
Printer.Print "Hello", "World"
Printer.Print "Hello"; Tab(8); "World"
```

Для печати графических объектов используются методы PSet. Line и Circle объекта Printer.

```
Printer.Line (1,1)-(10,5),,B
```

Готовые графические изображения различных форматов можно выводить на печать с помощью метода PaintPicture. Синтаксис оператора имеет следующий вид:

```
object.PaintPicture picture, x1, y1, width1, height1,
x2, y2, width2, height2, opcode
```

где object – в который можно выводить изображение;

picture – выводимое изображение (обязательный параметр);

x1, y1 – значения (Single), показывающие начало координат в единицах ScaleMode на объекте с которого начинается вывод изображения (обязательный параметр);

width1, height1 – параметры указывающие ширину и высоту изображения в единицах ScaleMode;

x2, y2 – значения (Single), показывающие начало координат части выводимого изображения в единицах ScaleMode;

width2, height2 – параметры указывающие ширину и высоту части выводимого изображения в единицах ScaleMode;

opcode – используется только с bitmaps.

Для этого в операторе нужно указать имя элемента управления, обладающего свойством Picture, и, при необходимости, позицию и размер изображения:

```
Printer.ScaleMode = vbCentimeters
Printer.PaintPicture picPictPrinter.Picture,5,5
```

После направления всех данных на печать с помощью объекта Printer готовая страница пока еще находится в оперативной памяти. Для запуска процесса печати этот объект должен получить сообщение о том, что формирование этой страницы завершено. Для этого предназначен метод NewPage. После того, как все страницы сформируются, вызывается метод EndDoc, который направляет сформированный документ на принтер.

Для того чтобы отменить или прервать печать, используется метод KillDoc.

1.1.4.3 Процесс печати

Элемент управления CommonDialog позволяет использовать стандартные диалоговые окна Windows, среди которых есть диалоговое окно установки параметров принтера и печати. Элемент управления CommonDialog предоставляет ряд свойств для управления этим диалоговым окном.

Для отображения диалогового окна его свойству Action присваивают значение 5 или используют метод ShowPrinter.

С помощью этого диалогового окна пользователь может выбрать новый принтер по умолчанию, количество и номера печатаемых страниц и число копий.

В отличие от других стандартных диалоговых окон, большая часть параметров, устанавливаемых в этом окне, не анализируется программистом, а передается непосредственно системе печати. Программист должен анализировать только параметры, устанавливаемые в группе "Печатать".

Если свойству Flags присвоить значение cdlPDPageNums и задать значения свойств Min и Max, то с помо-щью свойств FromPage и ToPage можно определить значения полей Страницы С и По.

Значение по умолчанию в поле Копии \Число копий можно установить, воспользовавшись свойством Copies:

```
CommonDialog1.Flags = cdlPDPageNums Or cdlPDCollate Or
cdlPDNoSelection
CommonDialog1.Copies = 2
CommonDialog1.Min = 0
CornmonDialog1 .Max = 10
CommonDialog1.ShowPrinter
```

```
nCopies = CommonDialog1.Copies
nFromPage = CommonDialog1.FromPage
nToPage = CommonDialog1.ToPage
```

Если значение свойства PrinterDefault равно True, то установки, сделанные пользователем, будут использованы в качестве установок принтера по умолчанию.

Если значение свойства TrackDefault равно False, то объект Printer всегда будет указывать на тот же принтер, независимо от изменения настроек принтера по умолчанию в панели управления.

1.2 Порядок выполнения работы

1.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Visual Basic 6.0».
3. Открыть проект “ Standard.EXE ”.
4. Разработать проект для реализации алгоритма, приведенного в задании.
5. Проект должен содержать несколько форм и программный модуль.
6. Формы должны содержать необходимые элементы управления для реализации алгоритма и облегчения работы пользователя с программой. Для каждого элемента управления используйте всплывающую подсказку. На каждом шаге в строку состояния должна выводиться подсказка о дальнейших действиях пользователя.
7. Для выбора имени файла при вводе и выводе данных в файл используйте окно Common Dialog. Для этого подключите к разрабатываемому проекту компонент Microsoft Common Dialog Control 6.0. (Проект/Компоненты)
8. Программный модуль должен содержать процедуры, выполняющие необходимые действия, обусловленные вариантом задания, и процедуры ввода-вывода данных в файл.
9. При старте проекта ввести необходимые данные и провести их обработку.
10. Результаты сохранить в файле на диске. Сравните результаты вывода данных в файлах последовательного и прямого доступа, просмотрев их с помощью просмотровщика данных из TotalCommander или Блокнота.
11. При вводе данных из файла прямого доступа предусмотреть возможность выбора произвольных записей.
12. Исходные данные и результаты обработки необходимо занести в отчет (сделать распечатку файлов, сохраненных в результате работы программы).

13. Запустить программу на выполнение и отладить при необходимости.

1.2.2 Порядок выполнения

1. Создайте структуру данных, соответствующую варианту задания. Создайте форму для ввода значений структуры. При работе программы введите данные в разработанную структуру с помощью формы (Разработанные списки должны содержать не менее 5 записей). После ввода сохраните данные в файлах последовательного и прямого доступа. При сохранении данных в файле последовательного доступа используйте кодировку данных (Добавить или вычесть несколько байт из ASCII кодов данных. Не забудьте сохранить в первой или последней записи это количество байт в том же файле, чтобы его можно было декодировать).

2. Введите данные из файла (при необходимости декодируйте их) и выведите их на экран в виде списка в соответствующих окнах (TextBox или ListBox) но уже на другой форме.

3. При выборе некоторого значения все данные, относящиеся к этому элементу структуры, должны быть выведены в отдельное окно.

1.2.3 Варианты заданий

1. Список студентов Вашей группы с указанием экзаменационных оценок за прошлый семестр по соответствующим дисциплинам.

2. Список любимых кинофильмов с указанием названия фильма, года выпуска фильма на экран, режиссера и исполнителей (не более 2-х) главных ролей.

3. Список преподавателей и дисциплин, читаемых ими в прошлом и этом семестрах, с указанием контроля знаний в семестре (экзамен/зачет) и Вашей оценки по этой дисциплине (если экзамен/зачет уже был или результаты модульного контроля для текущего семестра).

4. Список друзей (товарищей) с указанием номеров их телефона и дат рождения.

5. Список просмотренных за прошлый месяц по телевизору программ с указанием даты, времени начала и тематики программы.

6. Список преподавателей кафедры с указанием читаемых ими дисциплин и номера семестра, в котором эта дисциплина читается.

7. Список разработанных Вами до настоящего времени алгоритмов с указанием их имен, а также количества процедур и функций в них.

8. Список магазинов вблизи Вашего дома с указанием какого типа магазин, как часто Вы его посещаете, когда были последний раз, цель посещения, что приобрели.

9. Список рекомендованной литературы за последние два семестра с указанием дисциплины и преподавателя. Характеристика книги должна включать: Автор, название, год издания.

10. Список любимых аудиозаписей с указанием названия, исполнителя, композитора, времени звучания записи.

11. Расписание занятий в этом семестре с указанием времени, дисциплины, преподавателя и аудитории.

12. Список лабораторных работ проведенных к настоящему времени по дисциплинам, читаемым в этом семестре, с указанием даты проведения работы и отметки о сдаче отчета.

13. Список студентов группы с указанием модульных оценок по соответствующим дисциплинам.

14. Список прочитанных книг за прошлый год. Характеристика книги должна включать: Автор, название, год издания, количество страниц.

15. Список школьных друзей (товарищей) с указанием дат их рождения, а также имен и отчеств их родителей.

16. Список просмотренных кинофильмов (указать, где – в кинотеатре, по телевизору, видеозаписи, через Интернет) за прошлый месяц (указать дату и время просмотра).

17. Список школьных учителей с указанием дисциплин, читаемых ими, и оценок в Вашем аттестате (за последние два года обучения в школе).

18. Список преподавателей и читаемых ими дисциплин за прошедшие семестры, с указанием номера семестра, в котором читалась дисциплина.

19. Список соседних с Вашим домом улиц с указанием номеров ближайших домов.

20. Список друзей с пожеланиями им соответствующих благ, а также даты и времени приведения приговора в исполнение.

1.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы
2. Основные сведения о теме работы
3. Алгоритм разработанного проекта.
4. Перечень форм и элементов управления
5. Процедуры обработки событий и данных
6. Исходные данные и результаты их обработки

1.3 Контрольные вопросы

1. Перечислить последовательность этапов процесса работы с файлами.
2. Опишите связь приложения с файлом. Каково назначение функции FreeFile?
3. Какие существуют типы доступа к файлам?

4. Какие символы или константы Visual Basic используются для форматирования текста в файлах последовательного доступа?

5. Каким образом задается способ открытия файлов последовательного доступа?

6. С помощью каких операторов осуществляется чтение данных из файлов последовательного доступа?

7. С помощью каких операторов осуществляется запись данных в файлы последовательного доступа?

8. Назначение оператора Close?

9. Опишите отличие файлов прямого и последовательного доступа?

10. Опишите формат оператора открытия файла прямого доступа.

11. Опишите операторы ввода и вывода данных в файлы прямого доступа.

ЛАБОРАТОРНАЯ РАБОТА 2

ТИПЫ ОШИБОК. ИНСТРУМЕНТЫ ОТЛАДКИ ПРОЕКТОВ. РАБОТА С ТРЕХМЕРНОЙ ГРАФИКОЙ.

Цель работы – Научиться исправлять ошибки в программах, написанных на языке программирования Visual Basic 6.0. Научиться обрабатывать ошибки, возникающие в ходе работы программы. Научиться использовать инструменты отладки Visual Basic 6.0 при разработке проекта. Научиться использовать соглашения по именам переменных и структурное форматирование кода при написании программ на языке программирования Visual Basic 6.0. Научиться работать с трехмерной графикой.

2.1 Общие сведения

2.1.1 Типы ошибок

2.1.1.1 Синтаксические ошибки

Причиной возникновения синтаксической ошибки могут быть неправильно написанные ключевые слова, ошибки применения разделителей или недопустимые комбинации операторов. Visual Basic распознает синтак-сические ошибки сразу же после того, как курсор покидает эту логическую строку. Логическая строка может состоять из нескольких физических строк, разделенных символом подчеркивания (_). При обнаружении ошибки Visual Basic выдает сообщение с подробным пояснением ошибки. Такие сообщения достаточно информативны и позволяют легко определить причину возникновения ошибки и устранить ее.

Проверку синтаксиса можно включить или отключить с помощью опции Auto Syntax Check вкладки Ed-itor диалогового окна Tools\Options. Отключать проверку синтаксиса имеет смысл только в тех редких случаях, когда строка кода формируется путем копирования готовых фрагментов из других мест программы. В большинстве случаев отключать проверку синтаксиса не следует. Строка с синтаксической ошибкой выделяется красным цветом. Повторная проверка синтаксиса проверенных строк кода выполняется только после внесения в них изменений.

В Visual Basic 6.0 встроены средства, которые позволяют не только обнаружить синтаксическую ошибку, но и избежать ее при написании кода. Это, в частности, механизм контекстной подсказки или QuickInfo. QuickInfo — это небольшое окно, похожее на окно ToolTips, в котором автоматически отображается полный синтаксис вводимого оператора. Благодаря окну QuickInfo программист всегда имеет перед собой список аргументов процедуры.

Для уменьшения количества ошибок при написании имен, свойств и методов объектов, а также полей структур Visual Basic автоматически

отображает список доступных элементов. Вы можете выбрать элемент из списка либо ввести его имя с клавиатуры. В процессе ввода указатель списка автоматически перемещается к нужному элементу. Тип элемента списка (свойство или метод) указывает пиктограмма рядом с именем. Выбрать нужный элемент в списке можно также с помощью клавиш управления курсором. Нажатием клавиши [Tab] выделенный элемент вводится в текущую строку, причем текстовый курсор остается в этой строке. Для ввода выбранного элемента и перехода на следующую строку следует нажать клавишу [Enter].

Этот же список можно вызвать, воспользовавшись кнопкой List Properties/ Methods панели инструментов Edit или комбинацией клавиш [Ctrl+J]. Список констант открывается кнопкой List Constants или комбинацией клавиш [Ctrl+Shift+J].

Аналогично автоматическому списку элементов действует и функция дополнения слова. Если в окне кода введено несколько начальных символов свойства, метода или элемента структуры, которых достаточно для их однозначной идентификации, Visual Basic может дополнить недостающие символы. Для этого следует щелкнуть на кнопке Complete Word панели инструментов Edit или нажать клавиши [Ctrl+Пробел].

Дополнительные возможности при написании и отладке программы предоставляет цветовая кодировка элементов кода. Visual Basic позволяет выделять различным шрифтом и цветом фрагменты кода. Задать параметры шрифта и цвета можно на вкладке Editor Format диалогового окна Tools\Options.

При вводе кода Visual Basic автоматически устанавливает расстояние между отдельными словами. Например, возле знака равенства автоматически вставляются пробелы:

При написании программы не стоит полагаться на то, что Visual Basic сам правильно расставит все пробелы. Например, могут возникнуть сложности при использовании символа коммерческого И, или амперсанда (&). Он может применяться как соединитель строки (в таком случае он отделяется пробелами) или же, как идентификатор переменных типа Long, – используется без пробелов:

```
Label1.Caption = Numbers& & "Штук; Номер: " & &H100&
```

Обратите внимание, что в данном примере символ амперсанда (&) выполняет три различные функции. Сначала он служит идентификатором переменной Number типа Long, затем выступает как оператор соединения строк и, наконец, как разделитель для шестнадцатеричных чисел.

2.1.1.2 Ошибки в структуре программы

Ошибки в структуре программы — это синтаксические ошибки в многострочных операторах цикла и ветвления. Такие ошибки образуют отдельную группу ошибок, так как не распознаются Visual Basic при вво-

де. Однако при компиляции программы распознавание ошибки такого типа не представляет большой проблемы. В этом случае Visual Basic распознает такой не завершенный многострочный оператор, выдает сообщение об ошибке и выделяет ошибочный оператор.

Приложение не компилируется полностью, если его запускают из среды разработки нажатием клавиши [F5] или щелчком на кнопке Run панели инструментов. В этом случае ошибки в структуре программы на этапе выполнения выявляются только при обращении к процедуре, содержащей ошибочную структуру. Если же запуск программы осуществляется с помощью команды Start With Full Compile меню Run или нажатием [Ctrl+F5], то все ошибки в структуре программы обнаруживаются сразу при компиляции проекта. При компиляции Visual Basic также определяет имена объектов, не связанных с элементами управления, и выявляет переменные, которые не были явно объявлены (если была установлена ОПЦИЯ Option Explicit).

Проверка на отсутствие синтаксических ошибок и ошибок в структуре программы осуществляется и при создании выполняемого файла (команда Make *.EXE меню File).

2.1.1.3 Ошибки, возникающие при выполнении программы

В идеальном случае программа не должна бороться с ошибками в период выполнения. Однако разработчик должен предусмотреть вероятность появления сбойных файлов, переполнения памяти или ввода пользователем некорректных данных. Все это может послужить причиной возникновения ошибок при выполнении программы (Runtime Errors).

При обнаружении такой ошибки Visual Basic выводит соответствующее сообщение и приостанавливает выполнение программы. Если приложение было запущено из среды разработки, то существует возможность переключиться в режим отладки с помощью кнопки Debug либо в режим проектирования с помощью кнопки End.

Среда разработки относительно "мягко" реагирует на ошибки периода выполнения. Если же такая ошибка возникает после запуска выполняемого EXE файла, то приложение немедленно закрывается. Хотя сообщение об ошибке и появляется, перейти в режим отладки не возможно.

2.1.2. Инструменты отладки (Debugging Tools)

2.1.2.1 Режим отладки

Набор команд меню Run и назначение многих кнопок панели инструментов зависит от состояния среды разработки. В режим проектирования приложение можно только запустить, все же остальные возможности не доступны. При запуске можно выбрать один из двух вариантов: без полной компиляции или полную компиляцию всех процедур. Приложение

запускается нажатием клавиши [F5] или кнопки Start в среде разработки Visual Basic.

Переход в режим отладки выполняется нажатием клавиш [Ctrl + Break] или щелчком на кнопке Break. В режиме отладки можно выбирать один из вариантов: продолжать программу или перейти в режим разработки. В режим выполнения можно перейти, нажав повторно клавишу [F5] или щелкнув на кнопке Continue. Обратите внимание, что в режиме отладки кнопка Start носит название Continue. Название текущего режима отображается в квадратных скобках в строке заголовка Visual Basic. В режим отладки вы попадаете и тогда, когда во время выполнения программы, запущенной из среды разработки, возникла необрабатываемая ошибка выполнения. Большое преимущество режима отладки заключается в том, что выполнение программы приостанавливается в месте возникновения ошибки. Другим важным моментом является то, что при этом сохра-няются значения всех текущих переменных. В среде разработки Visual Basic инструменты поиска ошибок объединены в меню Debug.

2.1.2.2. Точка останова

Visual Basic предоставляет еще одну возможность переключения приложения в режим отладки. Это возможно благодаря точке останова (Breakpoint). Точка останова — это выделенная строка программы, на которой автоматически останавливается выполнение программы. По достижении этой строки программы Visual Basic также переходит в режим отладки. Имеется возможность установки и удаления точек прерывания в программе — с помощью полоски индикатора. Для отображения полосы индикатора следует установить опцию Margin Indicator Bar во вкладке Editor Format диалогового окна Tools\Options.

Установить удалить точки останова можно также с помощью контекстного меню или кнопки Toggle Breakpoint панели инструментов. Точки останова можно поместить в любой строке кода, включая заголовок процедуры (Sub/Function/Property) и строку End. Точки останова нельзя установить только в строках комментариев или пустых строках.

На панели инструментов Debug находится кнопка Toggle Breakpoint, позволяющая установить или удалить точку останова на текущей строке. Это можно сделать также нажатием клавиши [F9]. Установку или удаление точки останова для текущей строки можно выполнить с помощью команды Toggle Breakpoint меню Debug. Удалить все точки останова во всем проекте можно с помощью команды Clear All Breakpoint меню Debug. Установленные в среде разработки точки останова не сохраняются вместе с программой и не включаются в EXE файл при его создании.

Использование оператора Stop аналогично установке в программе точки останова. Если этот оператор встречается в программе, то Visual Basic переключается в режим отладки. Однако этот оператор целесооб-

разно использовать только при разработке приложения. В EXE файлах он выполняет действие, аналогичное оператору End, т.е. приводит к немедленному завершению программы.

Даже сама установка точки останова может помочь при отладке программы. Например, если точка установлена на заголовке процедуры, но переход в режим отладки не произошел, это значит, что данная процедура не вызывается при выполнении.

2.1.2.3. Следующий оператор

В режиме отладки Visual Basic особым образом выделяет строку, которая должна выполняться следующей. Сама строка выделяется желтым цветом, а на полосе индикатора рядом с ней появляется желтая стрелка.

Если выполнение программы прерывается в точке останова, то оба выделения комбинируются. При этом важно, что строка с точкой останова выделяется и как следующий оператор для выполнения, т.е. эта строка еще не выполнялась, а только подлежит обработке. Чтобы продолжить выполнение программы с любой другой строки, необходимо желтую стрелку полосы индикатора просто перетащить мышью на нужную строку. Если попытаться установить желтую стрелку на строку, которая не может быть выполнена, курсор мыши примет вид, указывающий на невозможность переноса.

Если при просмотре программы вы потеряли из виду текущую строку выполнения, то с помощью команды меню Debug\Show Next Statement можно вернуть ее в поле зрения в окне кода.

Маркирование следующей выполняемой строки позволяет получить различную информацию. Если после возникновения ошибки выполнения вы переходите в режим отладки, то маркировка показывает строку, в которой возникла ошибка.

Особый интерес представляет выделение следующей выполняемой строки при пошаговом выполнении. В этом случае можно точно проследить очередность выполненных операторов, что важно, например, в операторах ветвления, когда необходимо точно установить, какая ветвь программы выполняется.

Полезным может быть перенос следующей выполняемой строки. При изменении значения переменной можно проверить правильность выполнения программы при новом значении переменной. Для этого не нужно заново запускать приложение — достаточно повторить нужную часть кода.

2.1.2.4. Пошаговое выполнение программы

Если программа находится в режиме отладки, то она будет работать медленнее, так как все строки выполняются пошагово. При этом можно непосредственно наблюдать за результатами выполнения каждой строки.

Пошаговое выполнение является важным средством поиска ошибок и отладки программы. Существует несколько различных команд пошагового выполнения. Команды пошагового выполнения можно вызвать из меню Debug либо из панели инструментов Debug.

При пошаговом выполнении строки кода выполняются одна за другой. После выполнения одной строки кода маркер следующей строки перемещается на одну строку.

Шаг с заходом (команда Step Into) позволяет не только выполнить соответствующий оператор. Если это оператор вызова процедуры или функции, он дает возможность перейти в эту процедуру.

В результате нажатия кнопки Step Into в режиме разработки программа также переходит в режим выполнения. Visual Basic не может самостоятельно решить, какую из многочисленных процедур обработки события следует выполнить, поэтому после возникновения события, для которого существует процедура обработки, Visual Basic перейдет в режим отладки.

Шаг с обходом подобен шагу с заходом. Различие проявляется только при вызове текущей процедурой других процедур. Если при шаге с заходом осуществляется переход в вызываемую процедуру, то шаг с обходом выполняет вызов процедуры как единичный оператор, т.е. без захода. Шаг с обходом выполняется нажатием кнопки Step Over на панели инструментов Debug или комбинации клавиш [Shift+F8].Этот вид пошагового выполнения представляет интерес при поиске ошибки в процедурах, содержащих вызовы других процедур. Сначала можно протестировать текущую процедуру без захода в вызываемые. Если же выяснится, что ошибка возникает в вызываемой процедуре, то при следующем проходе следует войти в эту процедуру.

Команда Step Out меню Debug позволяет выполнить оставшуюся часть текущей процедуры и возвратиться в точку вызова.

Если текущая строка находится в вызванной процедуре, то с помощью команды Step Out остальная часть процедуры не выполняется пошагово. Отличие команды Step Out от команды Continue состоит в том, что после выхода из процедуры переключение в режим выполнения не происходит, если эта процедура была вызвана другой. Если же текущая процедура не была вызвана другой процедурой, то происходит переход в режим выполнения Visual Basic ожидает возникновения события, выполнение процедуры обработки которого начнется в режиме отладки.

Команда Run To Cursor меню Debug позволяет выполнить программу от текущей выполняемой строки до строки с установленным в ней текстовым курсором. Если текстовый курсор находится в выполняемой строке, то результат выполнения этой команды будет таким же, что и ко-

манды Continue. Для вызова команды Run To Cursor используется комбинация клавиш [Ctrl+F8].

2.1.2.5. Список вызовов

При поиске ошибок часто нужно знать последовательность вызова процедур. В окне Call Stack отображается список имен всех выполняемых в данный момент процедур. Первым отображается имя текущей процедуры. За ним следует список процедур в той последовательности, в которой они были вызваны. Имя процедуры обработки события указывается в конце списка. Таким образом, образуется список всех вызванных процедур Sub, Function или Property. После завершения процедура удаляется из списка.

Окно Call Stack позволяет отобразить команда Call Stack... меню View, которая доступна только в режиме отладки. Для открытия окна можно воспользоваться также комбинацией клавиш [Ctrl+L] или соответствующей кнопкой на панели Debug.

С помощью кнопки Show этого окна осуществляется переход в окно кода к выбранной в списке процедуре. Кроме этого, на полосе индикатора зеленым треугольником отмечается строка, содержащая вызов процедуры.

2.1.2.6. Отображение значений

Кроме контроля хода выполнения программы важной задачей инструментов отладки Visual Basic является проверка значений выражений. Для реализации механизма просмотра (Watch) Visual Basic предлагает несколько способов.

Контроль значений возможен только в режиме отладки. Более того, контролируемое выражение доступно только в определенных местах, например, значение локальной переменной можно проверить только в процедуре, в которой она объявлена. При попытке проверить значение выражения за пределами области определения появляется сообщение "Out of context"("Вне контекста").

Самый простой вариант просмотра значения переменной или выражения — использование окна Data Tips. Для открытия этого окна достаточно установить курсор мыши на соответствующем выражении в окне кода.

В режиме отладки можно также использовать команду QuickInfo меню Edit, которая позволяет отобразить синтаксис для переменной, функции, оператора, метода или процедуры, выбираемых в окне кода. Вызвать эту команду, действующую в режиме проектирования, можно также с помощью комбинации клавиш [Ctrl + I].

Команда ParameterInfo меню Edit позволяет получить информацию о параметрах используемой функции или оператора. Команда QuickInfo

отображает описание текущей функции, в то время как ParameterInfo отображает описание функции, которая является параметром процедуры.

Эта функция активизируется командой ParameterInfo меню Edit. При этом текстовый курсор должен находиться в контролируемом выражении. Для активизации функции можно также использовать комбинацию клавиш [Ctrl + Shift + I].

Еще одну возможность просмотра значений выражений предоставляет сохранившееся из предыдущих версии окно Quick Watch, вызываемое командой Quick Watch меню Debug, либо комбинацией клавиш [Shift+F9]. При вызове команды текстовый курсор должен находиться внутри имени контролируемой переменной. Для открытия окна можно также воспользоваться соответствующей кнопкой панели инструментов Debug.

Часто при поиске ошибки необходимо постоянно контролировать значения выражения. В этом случае, вместо того чтобы открывать каждый раз окно для просмотра значения, гораздо проще удобнее постоянно видеть значение выражения на экране. Для этого предназначена команда Add Watch...меню Debug. После выполнения этой команды отображается диалоговое окно Add Watch, позволяющее не только добавить нужное выражение в окно просмотра, но и определить дополнительные параметры просмотра и выполнения программы.

Диалоговое окно Add Watch позволяет внести изменения в контролируемое выражение. После внесения изменений соответствующее выражение появляется в окне просмотра.

Значения параметров, устанавливаемые при добавлении выражения в окно контрольного значения, можно изменить с помощью команды Edit Watch... меню Debug. В результате выполнения этой команды отображается диалоговое окно Edit Watch, похожее на окно Add Watch.

Это же окно можно вызвать с помощью комбинации клавиш [Ctrl + W]. Оно также позволяет не только редактировать, но удалять контролируемые выражения (кнопка Delete).

Рассмотренный механизм контроля значений переменных имеет большое значение. При поиске ошибок разработчику требуется как можно больше информации. Благодаря информации о текущем состоянии программы он легко может выявить источник ошибки.

2.1.3 Окна режима отладки.

2.1.3.1 Окно контрольного значения

Как уже упоминалось, после вызова команды Add Watch... или Edit Watch... меню Debug открывается окно контрольного значения со списком контролируемых выражений. Это окно открывается с помощью команды меню View\Watch Window.

Проще всего для добавления выражения в список воспользоваться методом Drag&Drop для перетаскивания выражения из окна кода в окно контрольного значения. В первом столбце окна отображается контролируемое выражение пиктограмма, отражающая способ просмотра контролируемого выражения, который устанавливается в диалоговом окне Add Watch.

Пиктограмма, изображающая очки (тип просмотра Watch Expression), показывает, что будет отображаться только текущее значение выражения, которое будет автоматически обновляться при переходе в режим отладки. Пиктограмма рук со знаком равенства (тип просмотра Break When Value Is True) показывает, что если значение контролируемого выражения становится равным True или ненулевым, то осуществляется автоматический переход в режим отладки. Пиктограмма руки с треугольником (знак Δ (дельта), или символ приращения) обозначает, что выполнение программы прерывается при изменении значения выражения.

Во втором столбце окна отображаются текущие значения выражений. Обратите внимание, что текущее значение отображается только в режиме отладки. В этом режиме можно изменять значение выражения (если это допускается). Для этого достаточно щелкнуть на нужном значении, а затем внести изменения.

Третий столбец окна отображает тип данных соответствующего выражения. Обратите внимание, что хотя сравниваются значения двух переменных разного типа, например, Single и Integer, типом данных результата является Boolean, так как выражение может возвращать только True или False.

В последнем столбце окна указывается объект, которому принадлежит выражение или переменная. Это значит, что значение отображается, если при просмотре область действия указанного объекта больше или равна текущей области действия.

2.1.3.2 Окно локальных переменных

Окно локальных переменных функционирует аналогично окну контрольного значения. Однако если в окно контрольного значения необходимо явно добавлять выражения, то в окне локальных переменных все локальные переменные отображаются автоматически.

Для открытия этого окна следует вызвать команду Locals Window меню View или щелкнуть на соответствующей кнопке панели инструментов Debug.

В первом столбце окна отображаются имена локальных объектов переменных. В первой строке приводится главный объект (форма, модуль или модуль класса), т.е. объект, которому принадлежит выполняемая процедура или функция. Во втором столбце окна выводятся текущие значения, которые можно редактировать, если они не защищены от записи. В

третьем столбце окна указывается тип данных соответствующей переменной.

2.1.3.3 Окно отладки

Для открытия окна отладки следует вызвать команду Immediate Window меню View ли воспользоваться комбинацией клавиш [Ctrl + G]. В этом окне можно не только изменять содержимое переменных или свойств, но и применять методы объектов, что позволяет имитировать логическую ошибку или вызывать процедуру. Для выполнения оператора нужно перейти на новую строчку клавишей [Enter]; текстовый курсор при этом может не находиться в конце строки.

После выполнения строка из окна не удаляется, поэтому ее можно выполнять несколько раз с измененными, при необходимости, значениями. В окне отладки можно осуществлять не только ввод, но и вывод, воспользовавшись методом Print. Для этого оператору Print передается требуемое выражение, и после нажатия клавиши [Enter] результат отобразится в следующей строке.

Окно отладки можно использовать и по-другому. В этом случае, используя объект Debug его метод Print, сообщения окну посылают из программного кода. Возможность такого вывода в окно отладки имеет несколько преимуществ.

Во-первых, выводить контрольные значения можно не только в режиме отладки, но и в режиме выполнения, что позволяет выводить выражения, не останавливая выполнение программы. Во-вторых, выведенные значения, отображаемые в окне отладки, можно просмотреть даже после остановки программы. Это важно в случаях, когда поведение программы при пошаговом выполнении отличается от поведения при нормальном выполнении, например при передаче фокуса или при приеме данных в режиме реального времени.

Объект Debug является системным объектом, и поэтому ключевое слово Debug нельзя использовать для задания имен других объектов.

Следует помнить, что при частом использовании окна отладки для вывода информации может замедлиться выполнение программы. Однако после удаления операторов Debug.Print или после создания EXE файла скорость выполнения восстанавливается.

2.2 Порядок выполнения работы

2.2.1 Задания для выполнения работы

1.Разработать алгоритм построения изображения трехмерных геометрических объектов, который необходимо занести в отчет.

2.Включить компьютер.

3.Запустить на выполнение программу «Visual Basic 6.0».

4.Открыть проект Standard.EXE.

5.Разработать формы, содержащие необходимое количество элементов управления для реализации алгоритма и облегчения работы пользователя с программой.

2.2.2 Порядок выполнения

1. Для каждого элемента управления используйте всплывающую подсказку. На каждом шаге в строку состояния должна выводиться подсказка о дальнейших действиях пользователя.
2. Написать коды обработки событий, связанных с элементами управления. Задание для каждого студента приведено ниже.
3. Запустить программу на выполнение.
4. При старте проекта ввести необходимые данные и провести их обработку (построить трехмерное изображение).
5. Для выбора имени файла при вводе и выводе данных в файл используйте окно Common Dialog. Для этого подключите к разрабатываемому проекту компонент Microsoft Common Dialog Control 6.0. (Проект/Компоненты)
6. Программный модуль должен содержать процедуры, выполняющие необходимые действия, обусловленные вариантом задания, и процедуры ввода-вывода данных в файл.
7. Сохранить созданное изображение в файле на диске в формате .bmp, чтобы затем просмотреть изображение в программе “Paint”. При сохранении изображения, созданного в PictureBox, воспользуйтесь оператором SavePicture picture, stringexpression, где picture – объект, содержащий изображение, а stringexpression – строка, в которой указан полный путь с именем файла, в котором необходимо сохранить изображение.

2.2.3 Варианты заданий

2.2.3.1 Построение изображений объектов в трехмерном пространстве

Дать двумерные изображения трехмерных геометрических объектов в диметрии. При построении трехмерных изображений в диметрия масштаб по осям z и y одинаков, а по оси x в 2 раза меньше; угол между осями y и z равен 90°; угол между осями x и y составляет 135°.

При разработке алгоритма программы необходимо получить решения соответствующих уравнений пересечения линий, плоскостей или поверхностей. Коэффициенты уравнений, описывающих линии, плоскости и поверхности вводить из текстовых окон. Используя уравнения линий, плоскостей и поверхностей определить необходимые для построения точки, уравнения линий и поверхностей и построить точки, линии или поверхности пересечения соответствующих объектов. Соответствующие уравнения можно взять из справочника по математике.

Изображения линий пересечения объектов и плоских граней тел должны быть окрашены в различные цвета.

2.2.2.2 Варианты заданий

Вариант № 1. Построить изображение прямой и окружности, центр которой лежит на прямой, а плоскость, в которой лежит окружность, перпендикулярна прямой. Показать след пересечения прямой и плоскости, в которой лежит окружность. Построить след пересечения плоскости, в которой лежит окружность, или прямой с одной из координатных плоскостей (прямая, плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 2. Построить изображение плоскости в виде треугольника, и другой плоскости, параллельной данной и находящейся на заданном расстоянии от первой (изобразить эту плоскость в виде четырехугольника). Построить след пересечения плоскости с одной из координатных плоскостей (плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 3. Построить изображение плоскости в виде треугольника, и другой плоскости, перпендикулярной данной и проходящей через прямую, лежащую в первой плоскости (изобразить эту плоскость в виде четырехугольника). Построить след пересечения плоскости или прямой с одной из координатных плоскостей (прямая, плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 4. Построить изображение плоскости в виде треугольника, и другой плоскости, перпендикулярной данной и проходящей через прямую, не лежащую в первой плоскости (изобразить эту плоскость в виде четырехугольника). Построить след пересечения плоскости или прямой с одной из координатных плоскостей (прямая, плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 5. Построить изображение в пространстве равнобедренного треугольника, не лежащего ни в одной из координатных плоскостей. Найти след его высоты на координатных плоскостях.

Вариант № 6. Построить изображение в пространстве квадрата, не лежащего ни в одной из координатных плоскостей. Найти следы его диагоналей на координатных плоскостях.

Вариант № 7. Построить изображение двух пересекающихся плоскостей, заданных в виде пересечения двух прямых каждая. Показать линию их пересечения. Построить след пересечения плоскости с одной из координатных плоскостей (плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее

обозначению).

Вариант № 8. Построить изображение плоскости в виде треугольника и прямой, проходящей через точку вне плоскости (треугольника) и перпендикулярную ей. Показать след пересечения прямой и плоскости. Построить след пересечения плоскости или прямой с одной из координатных плоскостей (прямая, плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 9. Построить изображение плоскости в виде треугольника. Показать след ее пересечения с одной из координатных плоскостей (по выбору – щелчок мышкой по изображению или ее обозначению).

Вариант № 10. Построить в трехмерном пространстве изображение прямой, проходящей через заданные две точки. Построить следы пересечения прямой с одной из координатных плоскостей (по выбору – щелчок мышкой по изображению или ее обозначению).

Вариант № 11. Построить в трехмерном пространстве изображение двух непересекаю-щихся прямых, одна из которых проходит через заданные две точки. Построить следы пересечения прямых с одной из координатных плоскостей (по выбору – щелчок мышкой по изображению или ее обозначению).

Вариант № 12. Построить в трехмерном пространстве изображение двух параллельных прямых, одна из которых проходит через заданные две точки, а другая смещена в заданном направлении на заданное расстояние. Построить следы пересечения прямых с одной из координатных плоскостей (по выбору – щелчок мышкой по изображению или ее обозначению).

Вариант № 13. Построить в трехмерном пространстве изображение двух перпендикулярных прямых, одна из которых проходит через заданные две точки, а вторая через заданную точку, не лежащую на первой прямой. Построить следы пересечения прямых с одной из координатных плоскостей (по выбору – щелчок мышкой по изображению или ее обозначению).

Вариант № 14. Построить в трехмерном пространстве изображение двух перпендикулярных прямых, одна из которых проходит через заданные две точки, а вторая через заданную точку, лежащую на первой прямой и проходящей в заданном направлении. Построить следы пересечения прямых с одной из координатных плоскостей (по выбору – щелчок мышкой по изображению или ее обозначению).

Вариант № 15. Построить в трехмерном пространстве изображение параболы и точки ее пересечения с координатными плоскостями.

Вариант № 16. Построить в трехмерном пространстве изображение прямой, заданной параметрически. Построить след пересечения плоско-

сти с одной из координатных плоскостей (плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 17. Построить изображение двух пересекающихся плоскостей, заданных в виде треугольников. Построить след пересечения плоскости с одной из координатных плоскостей (плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 18. Построить изображение плоскости в виде треугольника и прямой, не лежащей в плоскости. Показать след пересечения прямой и плоскости. Построить след пересечения плоскости или прямой с одной из координатных плоскостей (прямая, плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 19. Построить изображение отрезка прямой и плоскости в виде треугольника, проходящего через середину отрезка и перпендикулярного ему. Построить след пересечения плоскости или прямой с одной из координатных плоскостей (прямая, плоскость и координатная плоскость определяются по выбору – щелчок мышкой по изображению плоскости или ее обозначению).

Вариант № 20. Построить изображение спирали и точек ее пересечения с координатными плоскостями (задать радиус спирали, ее шаг и направление оси).

2.2.2.3 Построение трехмерных объектов

В двух графических окнах (PictureBox) дать двумерные изображения трехмерных геометрических объектов в двух проекциях: диметрия (масштаб по осям z и y одинаков, а по оси x в 2 раза меньше; угол между осями x и z равен 90°; угол между осями x и y составляет 135°) и изометрия (масштабы по осям x, y и z одинаковы; углы между всеми осями координат составляют 120°).

При разработке алгоритма задания используйте сведения из начертательной и аналитической геометрии (в случае возникновения трудностей обращайтесь за консультацией к преподавателю). Необходимые для построения параметры вводить из текстовых окон.

Изображения плоских граней фигур должны быть окрашены в различные цвета (используйте свойство FillColor объекта PictureBox).

Изображения цилиндрических или конических поверхностей делать линейчатыми с изменением цвета линий вдоль поверхности.

2.2.2.4 Варианты заданий

Вариант № 1. Построить изображение прямоугольной призмы с треугольным основанием (невидимые линии проводить пунктиром). Повернуть изображение на угол, указанный в режиме выполнения, вокруг

оси, перпендикулярной оси призмы, при нажатии кнопки “Поворот”.

Вариант № 2. Построить изображение эллипсоида вращения (координаты двух точек, через которые проходит ось вращения, задаются в режиме выполнения). При построении изображения можно использовать цилиндрические координаты. Построение проводить, используя метод Circle, в котором предусмотреть вывод эллипсов различных цветов, зависящих от координаты вдоль оси вращения эллипсоида.

Вариант № 3. Построить изображение кругового цилиндра (построение проводить, используя метод Line PictureBox, строить только видимую часть цилиндра). Повернуть изображение цилиндра так, чтобы его ось вращения проходила через центр основания цилиндра и точку вне его оси, указанную в режиме выполнения, при нажатии кнопки “Поворот”.

Вариант № 4. Построить изображение тетраэдра (невидимые линии проводить пунктиром). Повернуть изображение тетраэдра так, чтобы его ось симметрии проходила через центр основания тетраэдра и точку вне его оси, указанную в режиме выполнения, при нажатии кнопки “Поворот”.

Вариант № 5. Построить изображение пирамиды (невидимые линии проводить пунктиром), в основании которой находится квадрат. Повернуть изображение пирамиды так, чтобы её ось симметрии проходила через центр основания пирамиды и точку вне его оси, указанную в режиме выполнения, при нажатии кнопки “Поворот”.

Вариант № 6. Построить изображение конуса (построение проводить, используя метод Line PictureBox, строить только видимую часть конуса). Повернуть изображение конуса так, чтобы его ось симметрии проходила через центр основания конуса и точку вне его оси, указанную в режиме выполнения, при нажатии кнопки “Поворот”.

Вариант № 7. Построить изображение параллелепипеда (все непараллельные ребра разной длины, все углы $\neq 90°$, невидимые линии проводить пунктиром). Повернуть изображение параллелепипеда на угол, указанный в режиме выполнения, вокруг указанной оси координат, при нажатии кнопки “Поворот”.

Вариант № 8. Построить изображение конической поверхности (построение проводить, используя метод Line PictureBox, строить только видимую часть конуса), ось которой находится в фокусе параболы и перпендикулярна ее плоскости (три точки, через которые проходит плоскость, задаются в режиме выполнения), вершина конуса находится на расстоянии h от плоскости параболы, а сама поверхность проходит через параболу.

Вариант № 9. Построить изображение цилиндрической поверхности (строить только видимую часть цилиндра), ось которой находится в фокусе параболы и перпендикулярна ее плоскости (три точки, через ко-

торые проходит плоскость, задаются в режиме выполнения), а сама поверхность проходит через параболу (образующая цилиндра равна h).

Вариант № 10. Построить изображение однополостного гиперболоида (x2/a2 + y2/b2 – z2/c2 = 1). Построение организовать с помощью 3-х вложенных циклов по координатам z, x, y. Поверхность выводить с помощью точек (оператор: PictureBox.Pset (Хэкр, Yэкр), RGB(red(x), green(y), blue(z))).

Вариант № 11. Построить изображение двуполостного гиперболоида (x2/a2 – y2/b2 – z2/c2 = 1). Построение организовать с помощью 3-х вложенных циклов по координатам z, x, y. Поверхность выводить с помощью точек (оператор: PictureBox.Pset (Хэкр, Yэкр), RGB(red(x), green(y), blue(z))).

Вариант № 12. Построить изображение гиперболического параболоида (x2/p + y2/q = 2z; p>0, q>0). Построение организовать с помощью 3-х вложенных циклов по координатам z, x, y. Поверхность выводить с помощью точек (оператор: PictureBox.Pset (Хэкр, Yэкр), RGB(red(x), green(y), blue(z))).

Вариант № 13. Построить изображение параболоида вращения (координаты двух точек, через которые проходит ось вращения, задаются в режиме выполнения). Показать меридиональные сечения и сечения перпендикулярные оси вращения. При построении изображения можно использовать цилиндрические координаты.

Вариант № 14. Построить изображение гиперболоида вращения (координаты двух точек, через которые проходит ось вращения, задаются в режиме выполнения). Показать меридиональные сечения и сечения перпендикулярные оси вращения. При построении изображения можно использовать цилиндрические координаты.

Вариант № 15. Построить изображение конической поверхности, ось которой находится в фокусе гиперболы и перпендикулярна ее плоскости (три точки, через которые проходит плоскость, задаются в режиме выполнения), вершина конуса находится на расстоянии h от плоскости гиперболы, а сама поверхность проходит через гиперболу.

Вариант № 16. Построить изображение цилиндрической поверхности, ось которой находится в фокусе гиперболы и перпендикулярна ее плоскости (три точки, через которые проходит плоскость, задаются в режиме выполнения), а сама поверхность проходит через гиперболу (образующая цилиндра равна h).

Вариант № 17. Построить изображение октаэдра.

Вариант № 18. Построить изображение куба с вписанным шаром. Вырезать из изображения четверть.

Вариант № 19. Построить изображение двух пересекающихся под прямым углом цилиндров, радиус одного из которых в два раза больше

другого. Повернуть изображение так, чтобы ось симметрии одной из призм проходила через центр основания конуса и точку вне его оси, указанную в режиме выполнения, при нажатии кнопки “Поворот”.

Вариант № 20. Построить изображение прямоугольной призмы с квадратным основанием и пересекающего ее цилиндра, радиус которого равен ¾ стороны основания призмы. Вырезать из изображения четверть.

2.2.4 Содержание отчета

1. Цель работы.
2. Основные сведения о теме работы.
3. Алгоритм разработанного проекта.
4. Перечень форм и элементов управления.
5. Процедуры обработки событий, данных и ошибок.
6. Исходные данные и результаты их обработки.
7. Выводы.

2.3 Контрольные вопросы

1. Охарактеризуйте возможные типы ошибок, которые могут возникать при отладке и выполнении программы.
2. Что такое синтаксические ошибки? На каком этапе разработки проекта они выявляются? Как включить проверку синтаксиса?
3. На каком этапе разработки проекта выявляются ошибки в структуре программы?
4. Каковы действия среды разработки при обнаружении ошибок в период выполнения?
5. Как перейти в режим отладки программы?
6. Опишите назначение точек останова. Какие существуют способы установки и удаления точек останова?
7. Как маркируется в режиме отладки следующий выполняемый оператор?
8. Какую информацию можно получить, наводя маркер на тот или иной идентификатор переменной?
9. Опишите назначение пошагового выполнения программы.
10. Опишите назначение списка вызовов.
11. Перечислите и опишите назначение окон отладки.
12. Опишите назначение окна контрольного значения и отображаемой в нем информации.
13. Опишите назначение окна локальных переменных и отображаемой в нем информации.
14. Опишите назначение окна отладки, отображаемой в нем информации и возможности использования в нем оператора Print.
15. Опишите назначение стандарта имен при написании программ.

ЛАБОРАТОРНАЯ РОБОТА 3

ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ МЕТОДАМИ VISUAL BASIC. ИСПОЛЬЗОВАНИЕ MSFLEXGREED.

Цель работы – Создать проект построения графиков функций на Visual Basic 6.0, содержащий формы, различные элементы управления, процедуры и функции. Научиться работать с элементом управления MSFlexGreed. Научиться обрабатывать графические, текстовые и математические данные.

3.1 Общие сведения

3.1.1 Элемент управления MSFlexGreed

Элемент управления Microsoft FlexGrid (MSFlexGrid) отображает и позволяет оперировать с табличными данными. Он позволяет гибко сортировать, добавлять и форматировать таблицы, содержащие числовую, строковую и графическую информацию (picture). Если MSFlexGrid связан с элементом управления Data, то он отображает данные в режиме «только для чтения».

3.1.1.1 Свойства MSFlexGrid

В любой ячейке MSFlexGrid можно разместить текст, изображение (picture) или и то и другое. Свойства Row и Col определяют текущую ячейку в MSFlexGrid. Можно указать текущую ячейку в кодах VB или пользователь может изменить ее во время выполнения программы

Если текст слишком длинный для отображения в ячейке, то свойство WordWrap следует установить в True, при этом текст будет переноситься на другую строку в той же ячейке. Чтобы отобразить текст с переносами следует увеличить ширину колонки (свойство ColWidth) или высоту строки (свойство RowHeight).

Используйте свойства Cols и Rows, чтобы определить количество колонок и строк в MSFlexGrid.

Перед использованием MSFlexGrid в Вашем приложении, необходимо к проекту добавить файл MSFlxGrd.ocx. Для автоматического подключения файла к проекту поместите его в файл автозагрузки.

Свойства BackColor, BackColorBkg, BackColorFixed, BackColorSel позволяют устанавливать (возвращают) цвета различных частей MSFlexGrid.

```
object.BackColor [=color] ‘ синтаксис обращения [установки] свойства
```

Свойство CellAlignment возвращает или устанавливает горизонтальное и вертикальное выравнивание данных в текущей ячейке. Свойство недоступно на этапе разработки проекта.

```
object.CellAlignment [=value] 'синтаксис обращения [установки] свойства
```

Свойство CellBackColor устанавливает (возвращает) цвет заднего плана отдельной ячейки или диапазо-на ячеек, CellForeColor устанавливает (возвращает) цвет передего плана отдельной ячейки или диапазона ячеек. Свойства недоступны на этапе разработки проекта. Существуют и другие свойства управления цветом ячеек и др. объектов MSFlexGrid.

```
object.CellBackColor [=color] ' синтаксис обращения [установки] свойства
```

Свойства установки шрифтов в отдельной ячейке MSFlexGrid: CellFontBold; CellFontName; CellFontItalic; CellFontSize; CellFontUnderline; CellFontStrikeThrough; CellFontWidth; CellTextStyle.

Свойства CellHeight, CellLeft, CellTop, CellWidth – возвращают положение и размеры ячейки в твипах вне зависимости от установок свойства ScaleMode формы. Свойства недоступны на этапе разработки проекта.

Свойство CellPicture устанавливает (возвращает) изображение, отображаемое в текущей ячейке или в диапазоне ячеек. Свойство недоступно на этапе разработки проекта.

```
object.CellPicture [=picture]
```

Свойство ColSel устанавливает (возвращает) начало или конец колонки или диапазона колонок. Свойство RowSel устанавливает (возвращает) начало или конец строки или диапазона строк. Свойства недоступны на этапе разработки проекта. Чтобы выделить блок ячеек в MSFlexGrid, вначале следует установить свойства начала диапазона ячеек Col и Row, а затем свойства конца диапазона ячеек ColSel и RowSel.

Свойство FormatString устанавливает ширины колонок, выравнивание, фиксированный текст строки и колонки в MSHFlexGrid.

```
object.FormatString [= string]
```

На этапе разработки MSHFlexGrid анализирует и интерпретирует свойство FormatString, чтобы получить информацию: количество строк и колонок, текст для заголовков строк и колонок, ширины колонок и выравнивание в них.

Свойство FormatString содержит сегменты, разделенные символом «|». Текст между символами | определяет колонку и должен содержать специальные символы выравнивания. Эти символы устанавливают выравнивание в целой колонке по левому краю «<», по центру «^», или по правому краю «>». Кроме того, текст связывается с нулевой строкой по умолчанию, а ширина текста определяет ширину соответствующей колонки. Символ «;» в строке FormatString отделяет информацию о строках и может интерпретироваться как заголовки строк и высота строк. Текст присваивается по умолчанию строкам нулевой колонки. MSHFlexGrid создает дополнительные строки и колонки для согласования всех полей определенных FormatString. Лишние строки и колонки не уничтожаются, если

мало полей определено этим свойством. Чтобы удалить лишние строки и колонки используйте свойства Rows и Cols.

Свойство ScrollBars устанавливает или возвращает значение, показывающее имеет ли MSHFlexGrid полосы прокрутки и какие.

```
object.ScrollBars [=value]
```

The MSHFlexGrid имеет горизонтальную и вертикальную полосы прокрутки. (по умолчанию)

Свойство Sort устанавливает значение (value) для сортировки выбранных строк в соответствии с задан-ным критерием. Недоступно на этапе разработки.

```
object.Sort [=value]
```

Специальная сортировка. Используется событие Compare для сравнения строк.

По свойству Sort всегда сортируются целые строки. Для сортировки диапазона строк необходимо задать соответствующие значения свойствам Row и RowSel. Если Row и RowSel равны, то MSHFlexGrid будет сортировать все нефиксированные строки. Порядок сортировки определяется свойствами Col и ColSel. Сортировка всегда выполняется слева направо. Например, если Col = 3 и ColSel = 1, сортировка выполняется в последова-тельности колонок 1, затем 2, затем 3.

Свойство TextArray устанавливает или возвращает текстовое содержимое произвольной ячейки без изменения свойств Row и Col.

Свойство TextMatrix устанавливает или возвращает текстовое содержимое произвольной ячейки без изменения свойств Row и Col.

```
object.TextMatrix(rowindex, colindex) [=string]
```

(rowindex, colindex) – числовые выражения, определяющие значения индексов строк и колонок, на пересечении которых находится ячейка.

Свойство WordWrap устанавливает или возвращает значение, определяющее будет ли текст в ячейке записываться в одну строку, или будет переноситься, по словам в пределах ячейки.

```
object.WordWrap [=Boolean]
```

3.1.1.2. Методы MSFlexGrid

Метод AddItem добавляет строки в MSFlexGrid. Метод не поддерживает именованные аргументы.

```
object.AddItem(string, index, number)
```

Если MSHFlexGrid не содержит зон (связанных с неиерархическим набором записей), то BandNumber параметр необязательный. Если указано, то его значение должно быть нулем. Если свойство BandDisplay установлено как horizontal, а MSHFlexGrid связан с иерархическим набором записей, то BandNumber параметр обязателен. Если свойство BandDisplay установлено как vertical, то BandNumber параметр необходим только если зоны определены неявно. Параметр index всегда необязательный

Метод Clear очищает содержимое MSHFlexGrid (включая все тексты, изображения и форматирование ячеек). Метод не влияет на количество строк Rows и столбцов Cols.

```
object.Clear
```

Метод RemoveItem удаляет строки в MSFlexGrid. Метод не поддерживает именованные аргументы.

```
object.RemoveItem(index, number)
```

3.1.1.3. События MSFlexGrid

Событие Compare возникает, если свойство Sort MSHFlexGrid установлено в Custom Sort (9), так что пользователь может сам переделать процесс сортировки. Обработка этого события осуществляется процедурой, параметры котрой приведены в таблице 3.1:

```
Private Sub object_Compare(row1, row2, cmp)
```

Таблица 3.1 Параметры процедуры обработки события:

Part	Description
row1	Длинное целое указывающее первую из сравниваемых строк.
row2	Длинное целое указывающее вторую из сравниваемых строк.
cmp	Целое представляющее порядок сортировки каждой пары.

Обработка события должна сравнить строки row1 и row2 и установить cmp как это показано в таблице 3.2

Таблица 3.2 Значения cmp в зависимости от сравнения строк row1 и row2

Setting	Description
–1	Если *row1* может появиться перед *row2*.
0	Если обе строки равны или обе могут появиься перед другими.
1	Если *row1* может появиться перед *row2*.

Событие EnterCell возникает, когда текущая активная ячейка заменяется другой. Обработка этого собы-тия осуществляется процедурой:

```
Private Sub object_EnterCell()
```

Событие LeaveCell возникает, когда текущая активная ячейка заменяется другой. Обработка этого события осуществляется процедурой:

```
Private Sub object_LeaveCell()
```

Событие RowColChange возникает, когда текущая активная ячейка заменяется другой. Обработка этого события осуществляется процедурой:

```
Private Sub object_RowColChange()
```

3.1.2. Построение графиков функций

3.1.2.1. Ввод данных в процедуру.

При разработке процедуры построения графиков функций необходимо в первую очередь задаться, каким образом данные, выводимые на график, будут передаваться в процедуру. В случае, когда в процедуру передается один массив, данные в нем следует располагать следующим об-

разом (хотя возможны и другие варианты). Первый столбец содержит абсциссы первого графика, второй столбец – ординаты первого графика, далее в том же порядке для второго графика и т.д. Следовательно, массив данных, выводимых на график, содержит количество столбцов, равное удвоенному числу графиков, а количество строк равно максимальному количеству точек, выводимых на график. Количество точек, выводимых на каждом графике, должно при этом содержаться в нулевом (первом) элементе столбца абсцисс. Поэтому столбцы ординат данных, выводимых на график, в нулевом элементе не содержат никакой информации.

Следует заранее задаться количеством (масштабных) отметок по осям абсцисс и ординат.

3.1.2.2. Определение масштаба графика

Для определения масштаба графика необходимо определить максимальные и минимальные значения по осям графика, а затем округлить их до стандартных чисел.

Чтобы определить максимальные и минимальные значения по осям графика, необходимо для каждого из массивов с данными об абсциссах и ординатах точек графиков найти максимальное и минимальное значения (используйте разработанные ранее процедуры или функции). Затем среди этих значений (очевидно, их следует запоминать в массиве, скажем, «MiniMax»), также следует найти максимальное и минимальное значения по осям графика. Именно эти значения и будут максимальными и минимальными значениями, которые следует объявлять как глобальные переменные, по осям графика.

Цифровые масштабные отметки на осях графика проставляются напротив главных и промежуточных отметок на осях. Следует заранее определиться, сколько промежуточных отметок будет выведено на осях абсцисс и ординат.

Рассмотрим, каким образом, выбираются масштабы по осям. В процедуру передается параметр, значения которого указывают, выбирается ли масштаб по оси абсцисс или ординат. В зависимости от этого параметра внутренним переменным процедуры, соответствующим минимальному и максимальному значениям, присваиваются соответственно, значения xmin, xmax или ymin, ymax. Следующий параметр, передаваемый в процедуру выбора, указывает, выбирается ли число из стандартного рада ЕСКД для минимального (следует выбирать число меньшее, чем минимальное) или для максимального (следует выбирать число большее, чем максимальное)

Если абсолютная величина минимального значения меньше какого-либо малого положительного числа, например 10-15, то его значение обнуляется, а максимальное значение нормализуется. При этом определяется порядок числа N = Int(Log(Abs(x)) / Log(10)) и мантисса числа m = x /

10 ^ N. После этого мантисса округляется до ближайшего числа из ряда (1) или (2) в зависимости от того для какой оси определяется значение. Далее, если оказывается, что порядок числа N > 3 или N < -1, то формируется строка в виде " *10^" & Str$(N), которая выводится в обозначении единиц измерения для соответствующей оси, в противном случае для максимального значения будем иметь m * 10 ^ N. Аналогично обрабатывается ситуация, если абсолютная величина максимального значения меньше какого-либо малого положительного числа, например 10^{-15}.

Если абсолютная величина минимального значения меньше абсолютной величины максимального значения, то вычисляется величина $D_g = Abs(x_m / x_n)$, где x_m и x_n – максимальное и минимальное значения для соответствующей оси координат. В зависимости от значения этой величины можно выделить три случая:

• $D_g \geq 20$ – в этом случае заменяем минимальное значение нулем, а максимальное значение нормализуется.

• $D_g < 1.2$ – в этом случае нормализуется разность $D_g = x_m – x_n$, как максимальное значение. Затем минимальное значение находится как разность величин $x_n = x_m – D_g$.

• $1.2 \leq D_g < 20$ – в этом случае нормализуется максимальное значение и нормализуется разность $D_g = x_m – x_n$, как максимальное значение. Затем минимальное значение находится как разность нормализованных величин $x_n = x_m – D_g$.

Если абсолютная величина максимального значения меньше абсолютной величины минимального значения, то вычисляется величина $D_g = Abs(x_n / x_m)$. В зависимости от значения этой величины также рассматриваются те же три случая (только следует поменять местами x_m и x_n).

После определения масштаба по обеим осям координат и при необходимости степенных множителей и соответствующих строк, задается масштаб графика:

```
picGraPlot.Scale (xmin, ymax)-(xmax, ymin)
```

при условии, что picGraPlot – имя PictureBox’а, в котором строится график.

3.1.2.3. Вывод осей координат

Вывод осей координат на график осуществляется в цикле вместе с промежуточными линиями. Ширина линий (свойство DrawWidth PictureBox’а) сетки, как правило, выбирается меньшей, чем внешняя рамка графика. Одновременно выводятся числовые значения в соответствующие метки (Label), находящиеся вблизи линий сетки (числовые значения для меток определены при выборе масштаба для графика).

3.1.2.4. Вывод графика

При выводе графика в PictureBox организуется цикл по всем парам (xi, yi). В цикле методом Pset выводится точка с такими координатами. Затем вокруг этой точки строится значок (окружность, квадрат, ромб и т.п.), соответствующий этому графику. А затем выводимая точка и предыдущая точка графика соединяются отрезком прямой (график представляется в виде ломаной линии), или для более гладкого графика выводимая точка и предыдущая точка графика соединяются параболой (используются две предыдущие точки графика).

3.1.2.5. Легенда

Если на графике присутствует несколько кривых, то необходимо, во-первых, на каждой кривой выводить свои значки, а во-вторых, в нижней части графика должна быть помещена легенда, в которой указывается какой значок, соответствует названию той или иной выводимой зависимости. Как правило, для легенды используются элементы управления PictureBox и Label.

3.2 Порядок выполнения работы

3.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Visual Basic 6.0».
3. Открыть проект Standard.EXE.
4. Разработать форму, создав на ней необходимые элементы управления, для построения графиков и программный модуль, в котором будут создаваться массивы точек, выводимых на график в соответствии с индивидуальным заданием.
5. Отладить проект. Созданный проект сохранить для использования при выполнении лабораторных работ по курсу «Вычислительная математика» и ряду других дисциплин.

3.2.2 Порядок выполнения

1. Разработать проект для реализации варианта задания.
2. Проект должен содержать форму и программный модуль.
3. Форма должна содержать строку состояния, меню, панель инструментов и необходимые элементы управления для реализации алгоритма и облегчения работы пользователя с программой. Для каждого элемента управления используйте всплывающую подсказку (свойство ToolTeepText). На каждом шаге в строку состояния (StatusBar) должна выводиться подсказка о дальнейших действиях пользователя. При необходимости ProgressBar должен отражать ход выполнения вычислительного процесса и вывода графики в PictureBox.

4. Для ввода данных используйте окно InputBox. Окно MessageBox используйте для вывода сообщений и/или для выбора ветви хода вычислительного процесса.

5. Часть функций элементов управления продублируйте командами клавиатуры, для чего используйте события клавиатуры и анализ кода нажимаемых клавиш.

6. Используйте события мыши для выделения и/или изменения вида объектов, над которыми перемещается указатель мыши.

7. Программный модуль должен содержать процедуры, выполняющие необходимые действия, обусловленные вариантом задания.

8. При старте проекта ввести необходимые данные и провести их обработку.

9. Исходные данные и результаты обработки необходимо занести в отчет.

3.2.3 Варианты заданий

3.2.3.1 Общее задание для всех:

Построить геометрическую фигуру. Для чего:

• В программном модуле написать процедуру, вычисляющую координаты точек (концы отрезков прямых, центры окружностей, радиусы окружностей и т.п.), необходимых для выполнения построения.

• Значения координат и название соответствующего выводимого элемента хранить в MSFlexGrid (названия колонок: выводимый элемент; X; Y).

• Построить по рассчитанным точкам геометрическую фигуру.

Построить два графика функций. Для чего:

• В программном модуле написать функции, вычисляющие массивы значений координат точек, выводимых на график. Для удобства координаты точек можно рассчитывать в полярных координатах, а затем переводить в декартовы, или использовать параметрическое задание кривой.

• Значения координат точек графика вывести в MSFlexGrid (названия колонок: № п.п.; X; Y).

• Построить по рассчитанным точкам график функции.

3.2.3.2 Варианты задания 1 для выполнения проекта (построить геомет-рическую фигуру).

1. Равносторонний треугольник, вписанный в окружность.
2. Окружность, вписанная в квадрат.
3. Произвольный треугольник, вписанный в окружность.
4. Окружность, вписанная в равносторонний треугольник.
5. Правильный шестиугольник, вписанный в окружность.

6. Окружность и ромб, меньшая диагональ которого является диаметром окружности.

7. Прямоугольный треугольник, вписанный в окружность.

8. Окружность, вписанная в произвольный треугольник.

9. Прямоугольник, вписанный в окружность.

10. Окружность, вписанная в произвольный прямоугольник.

11. Ромб и окружность, диаметром которой является большая диагональ ромба.

12. Окружность, вписанная в правильный шестиугольник.

13. Квадрат, вписанный в окружность.

14. Окружность, вписанная в произвольную трапецию (окружность касается параллельных сторон трапеции).

15. Трапеция, вписанная в окружность (размеры сторон рассчитать так, чтобы все четыре вершины находились на окружности).

16. Окружность, вписанная в произвольную трапецию (окружность касается наклонных сторон трапеции).

17. Ромб и окружность, диаметром которой является одна из сторон ромба.

18. Окружность, диаметром которой является основание равнобедренного треугольника, высота которого в два раза больше основания.

19. Окружность, вписанная в равнобедренную трапецию (окружность касается наклонных сторон трапеции).

20. Окружность, вписанная в равнобедренную трапецию (окружность касается параллельных сторон трапеции).

21. Шестилепестковую розетку, лепестки которой образованы дугами окружностей, центры которых лежат на окружности такого же радиуса.

3.2.3.3 Варианты задания 2 для выполнения проекта (построить графики двух функций).

Номер варианта задания соответствует номеру студента в журнале группы.

№ студента в журнале	№№ кривых	№ студента в журнале	№№ кривых
1	27, 2	11	13, 22
2	18, 4	12	21, 14
3	3, 16	13	15, 19
4	1, 12	14	17, 2
5	8, 26	15	3, 11
6	5, 20	16	6, 18
7	10, 6	17	9, 20
8	7, 25	18	8, 1

9	11, 24	19	23, 7
10	23, 9	20	16, 10

Уравнения кривых, для построения графиков

1. Построить лемнискату Бернулли: $(x^2 + y^2)^2 – a^2(x^2 – y^2) = 0$ или $\rho^2 – a^2\cos(2\varphi) = 0$.

2. Построить циклоиду $\begin{cases} x = rt - d\sin t \\ y = r - d\cos t \end{cases}$; t – параметр.

3. Построить эпициклоиду $\begin{cases} x = (r+R)\cos t - r\cos\left[(r+R)\frac{t}{r}\right], \\ y = (r+R)\sin t - r\sin\left[(r+R)\frac{t}{r}\right]. \end{cases}$; t – параметр.

4. Построить розу $\rho = a\sin(k\varphi)$, где k = m / n; m, n – целые.

5. Построить астроиду: $x^{2/3} + y^{2/3} = a^{2/3}$ или $x = a*\cos^3(t)$, $y = a*\sin^3(\varphi)$.

6. Построить строфоиду: $y^2 = x^2\frac{d+x}{d-x}$; или $\rho = -d\frac{\cos 2\varphi}{\cos\varphi}$.

7.Построить эпитрохоиду: $\begin{cases} x = (R+mR)\cos mt - h\cos(t+mt), \\ y = (R+mR)\sin mt - h\sin(t+mt). \end{cases}$ (параметрическое задание кривой)

8. Построить крест: $x^2y^2 = a^2(x^2 + y^2)$ или $\rho = 2a/\sin(2\varphi)$.

9. Построить гипотрохоиду: $\begin{cases} x = (R-mR)\cos mt + h\cos(t-mt), \\ y = (R-mR)\sin mt - h\sin(t-mt). \end{cases}$ (параметрическое задание кривой)

10. Построить для разных значений m, d, a овал Декарта: $\sqrt{x^2+y^2} + m\sqrt{(x-d)^2+y^2} = a$ или $\rho + m\sqrt{\rho^2 - 2\rho d\cos(\varphi) - d^2} = a$.

11. Построить для разных значений c, a овал Кассини: $\left(x^2+y^2\right)^2 - 2c^2\left(x^2-y^2\right) = a^4 - c^4$ или $\rho^4 - 2\rho^2c^2\cos\left(2\varphi\right) = a^4 - c^4$.

12. Построить лемнискату Бута: $(x^2 + y^2)^2 + (2m^2 + n)x^2 + (2m^2 – n)y^2 = 0$ или $\rho^2 = a^2\cos^2(\varphi) – b^2\sin^2(\varphi)$ если $n > 2m^2$; и $\rho^2 = –a^2\cos^2(\varphi) + b^2\sin^2(\varphi)$ если $n < –2m^2$; ($a^2 = |2m^2 + n|$, $b^2 = |2m^2 – n|$).

13. Построить трисектрису: $y^2 = x^3(3a – x)/(a + x)$ или $\rho = a(4\cos(\varphi) – \cos^{-1}(\varphi))$.

14. Построить Декартов лист: $x = \frac{3at}{1+t^3}$, $y = \frac{3at^2}{1+t^3}$.

15. Построить кардиоиду: $(x^2 + y^2 – ax)^2 = a^2(x^2 + y^2)$ или $\rho = a(1 + \cos(\varphi))$.

16. Построить неоциду: $\rho = a\varphi + l$; a, $l > 0$.

17. Построить циссоиду Диоклеса: $y^2(a - x) = x^3$ или $\rho = a(\cos^{-1}(\varphi) - \cos(\varphi))$.

18. Построить кривую Штейгера: $(x^2 + y^2)^2 + 8rx(3y^2 - x^2) + 18r^2(x^2 + y^2) - 27x^4 = 0$.

19. Построить локон Аньези: $y(a - x^2) = a^3$, $a > 0$.

20. Построить кохлеоиду $\rho = a\frac{\sin(\varphi)}{\varphi}$.

21. Построить улитку Паскаля: $\rho = b - a*\cos(\varphi)$.

22. Построить спираль Архимеда: $\rho = a\varphi$.

23. Построить спираль Ферма: $\rho = a \times\left(\pm\sqrt{\varphi}\right)$.

24. Построить спираль Галилея: $\rho = a\varphi^2 - l$, $a, l > 0$.

25. Построить логарифмическую спираль $\rho = a^{\varphi}$.

26. Построить гиперболическую спираль $\rho = a / \varphi$.

27. Построить параболическую спираль $\rho = a \times\left(\pm\sqrt{\varphi}\right) + l$; $l > 0$.

3.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Ответы на контрольные вопросы.
3. Алгоритм (и/или сценарий) работы проекта.
4. Изображение таблицы данных в MSFlexGrid.
5. Изображение формы для вывода графиков с построенным графиком в соответствии с вариантом задания.

3.3 Контрольные вопросы

1. Описать назначение элемента управления MSFlexGrid.
2. Описать назначение свойств Row, Col, Cols и Rows, объекта MSFlexGrid.
3. Перечислить свойства установки цвета различных частей MSFlexGrid.
4. Описать назначение свойства CellAlignment объекта MSFlexGrid.
5. Описать назначение свойств TextArray и TextMatrix объекта MSFlexGrid.
6. Описать назначение метода AddItem объекта MSFlexGrid.
7. Описать назначение метода Clear объекта MSFlexGrid.
8. Описать назначение метода RemoveItem объекта MSFlexGrid.
9. Описать основные этапы построения графика функции.
10. Описать основные этапы определения мачштаба графика функции.

11. Описать процесс округления чисел до стандартных значений.
12. Назначение легенды на графике.

ЛАБОРАТОРНАЯ РОБОТА 4

МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ЗАВИСИМОСТЕЙ С ИСПОЛЬЗОВАНИЕМ VISUAL BASIC.

Цель работы – Научиться использовать соглашения по именам переменных и структурное форматирование кода при написании программ на языке программирования Visual Basic 6.0. Создать проект на Visual Basic 6.0, содержащий формы, различные элементы управления, процедуры и функции. Научиться обрабатывать графические, текстовые и математические данные.

4.1 Общие сведения

4.1.1 Присвоение имен переменным

Цель использования стандарта имен — сделать более понятным область действия, тип данных и назначение переменной. При присвоении имени часто используется общепризнанная "венгерская нотация". В соответствии с этой нотацией имя состоит из одного или нескольких префиксов и базового имени:

```
<Префикс><Базовое_имя>
```

При задании имен в Visual Basic действуют следующие ограничения:

- имя должно начинаться с буквы;
- имя может содержать только буквы, числа и символ подчеркивания;
- максимальная длина имен форм и элементов управления — 40 символов, имена переменных и типов могут иметь длину до 255 символов;
- в качестве имен нельзя использовать зарезервированные слова Visual Basic.

Одним из самых важных отличительных признаков переменной является ее тип данных, поэтому он должен отражаться в имени переменной. Обычно в качестве префикса используются трехзначные комбинации символов, хотя для наиболее часто используемых типов практичнее использовать только один символ.

Использование префиксного написания имеет то преимущество, что одинаковые типы данных можно записывать в алфавитной последовательности непосредственно один под другим.

Благодаря использованию префикса типа легко локализовать ошибки преобразования.

Использовать тип данных Variant следует осторожно. Во-первых, его внутренние функции преобразования работают медленнее, чем соответствующие функции Visual Basic (например, CCur и т.п.). Во-вторых, при неявных преобразованиях типа могут возникать каверзные ошибки.

Область действия переменной также должна отражаться в ее имени. Для этого используются префиксы, приведенные в таблице 4.1.

Таблица 4.1. Префиксы области действия

Префикс	Область действия
g	Глобальная
m	Модуль, форма, класс
отсутствует	Локальная

Обратите внимание, что локальные переменные не имеют собственного префикса, что отличает их от переменных модуля, класса, формы или глобальных.

Структуре <Область действия><Тип данных><Базовое имя> соответствует, на пример, mlRecordCount для переменной формы, модуля или класса типа Long с базовым именем RecordCount.

При задании базовых имен также следует соблюдать определенные правила. Имя переменной должно отражать ее содержимое. Только имена "i", "j" и "k" принимаются в качестве счетчика:

Избегайте непонятных сокращений типа vntADatum, vntActD или vntAD. При необходимости сокращения следует отказываться от гласных. Значительно повышает читабельность имен использование прописных строчных букв. При написании префикса строчными буквам понятно, где заканчивается префикс, а где начинается базовое имя.

В первых версиях Visual Basic константы записывались прописными буквам. Теперь для них используются как прописные, так строчные буквы (например, vbKeyF1). Но все же для констант, определяемых пользователем, рекомендуется применять прописные буквы:

```
Const PI = 3.14159265
```

Функции возвращают значения определенного типа, следовательно, имена функций также могут содержать сокращения для соответствующего типа данных. При передаче аргументов, наряду с типом данных, важное значение имеет вид передачи. Поэтому при передаче аргументов также может использоваться префикс, определяющий вид передачи аргумента — как ссылки (ByRef) или как значения (ByVal).

Имена присваивают не только переменным, но и объектам. И эти имена также должны отражать основ-ные особенности объекта.

В элементах управления формах используются правила, сходные с правилами присвоения имен для переменных и констант.

В приложениях часто используются меню. Для этого элемента управления применяются несколько отличные правила наименования. В

качестве префикса используется mnu, а имя должно отображать структуру меню. При использовании такого правила именования взаимосвязанные имена записываются одно под другим в алфавитном порядке, и каждое имя дополнительно отражает позицию команды в дереве меню.

Для того, чтобы имена были короче, можно использовать горячие клавиши вышестоящих уровней меню.

Иногда (хотя этого следует избегать) элементу управления присваивают имя, являющееся зарезервированным словом Visual Basic.

```
Select.Caption = "Hello"
```

В этом примере зарезервированное слово Select используется как имя объекта, что приводит к синтаксической ошибке. Если же обращаться к элементу управления с указанием его родителя (например, формы) либо как к внешнему объекту, заключив его имя в квадратные скобки, то такой доступ возможен:

Объекты базы данных являются важной составной частью Visual Basic, поэтому их также следует снабжать соответствующими префиксами.

Число приложений, обладающих OLE автоматизацией, стремительно растет. Поэтому становится трудно подбирать трехсимвольные префиксы.

Для того, чтобы найти определенный префикс, используются первые три символа имени объекта:

```
Dim rngRange As Range
Dim wkbMap As Workbook
```

Если используются OLE объекты из других приложений, желательно указывать происхождение объекта.

```
Dim wobApp As Object 'Microsoft WordBasic
Dim xlbApp As Object 'Microsoft ExcelBasic
```

В этом случае структура префикса также зависит от вида приложения среды, но решающим фактором всегда является последовательное использование установленных правил.

Кроме правильного назначения имен, существуют другие способы избежания проблем. К ним относятся, наряду с комментариями, явное объявление переменных и структурное форматирование кода.

Обычно в языках высокого уровня все переменные должны объявляться явно. Однако Visual Basic допускает использование не объявленных явно переменных. Для того чтобы не забыть объявить переменную, используйте опцию Option Explicit, добавляя ее в раздел (General)(Declaration) контейнера. Но для того, чтобы не делать это самим и не забыть это сделать, установите опцию Require Variable Declaration вкладки Editor диалогового окна Tools \Options.

После этого во всех вновь создаваемых контейнерах (формах, модулях, классах) в секции объявлений автоматически добавляется строка Option Explicit. Однако это не происходит в уже существующих контейнерах.

При написании кода операторы также следует располагать по определенным правилам. Конечно, это не жесткое требование и вы можете располагать операторы как угодно, однако, правильно структурированный код не просто легче воспринимается — это может помочь и при поиске ошибок.

```
Sub Test ()
  For i = iStart To iEnd
                Beep
                If ... Then
                              MsgBox "True"
                Else
                              MsgBox "False"
                End If
  Next i
End Sub
```

Благодаря приведенному в примере построению можно быстро обнаружить ошибку структуры. Синтаксическая конструкция заканчивается в том же столбце, в котором начинается.

Для разделения одной логической строки на несколько физических в Visual Basic существует специальная комбинация символов (Пробел +Символ подчеркивания), называемая разделителем. Использование разделителей может быть полезным для улучшения читабельности программы, особенно если некоторые длинные операторы не помещаются полностью на экране. Однако следует учитывать, что в одной логической строке может быть только 24 разделителя.

Благодаря использованию разделителей возможно форматирование кода таким образом, чтобы длинная строка полностью появлялась на экране, и ее можно было редактировать, не используя полосы прокрутки.

При использовании разделителя строк важным является пробел перед ним. Подчеркивание — единственный небуквенный и нецифровой символ, допустимый в имени, поэтому если он добавляется прямо к слову, то считается составной частью имени. Разделитель с пробелом перед ним распознается правильно. Однако слишком частое использование разделителей делает код программы слишком длинным, что, в свою очередь, может затруднить его просмотр.

Visual Basic позволяет не только разбивать одну логическую строку на несколько физических, но объединять в одной строке несколько операторов. Для объединения операторов в одну строку используется символ двоеточия (:). Аналогично разделителю строк, символ объединения строк

используется для повышения читабельности кода. Благодаря объединению простых операторов в одну строку, программа становится короче и на экране помещается большая часть кода.

В Visual Basic оператор +(плюс) может использоваться как для математического сложения, так и для соединения (конкатенации) строк. Поэтому при сложении лучше использовать символ +, а при соединении строк — символ &.

Кроме того, следует помнить, что символ & используется еще в качестве суффикса для переменных типа Long. Поэтому не нужно надеяться на автоматическую вставку пробела редактором Visual Basic, если оператор & присоединен непосредственно к имени.

Для выделения комментариев в Visual Basic используйте оператор Rem или символ '. Перед заголовком процедуры обычно помещается комментарий с описанием процедуры. Перед заголовком обычно описывается задача процедуры, а затем назначение используемых входных и выходных параметров. В конце оператора после символа можно добавить комментарий. Обратите внимание, что оператор Rem может добавляться в строку только после символа «:». Отдельные разделы в процедуре также можно снабжать комментариями.

4.2 Порядок выполнения работы

4.2.1 Задания для выполнения работы

1. Включить компьютер.
2. Запустить на выполнение программу «Visual Basic 6.0».
3. Открыть проект Standard.EXE.
4. Разработать форму, содержащую несколько элементов управления
5. Написать коды обработки событий, связанных с элементами управления. Задание для каждого студента приведено ниже. При написании кодов использовать соглашение по именам переменных и структурное форматирование кода. Использовать комментарии для указания информации, поясняющей назначение отдельных блоков программного кода, процедур и функций.
6. Запустить программу на выполнение.

4.2.2 Порядок выполнения

1. Разработать проект для реализации алгоритма, разработанного в соответствии с заданием. Проект должен содержать несколько форм и программный модуль. Формы должны содержать необходимые элементы управления для реализации алгоритма и облегчения работы пользователя с программой. Для каждого элемента управления используйте всплывающую подсказку. На каждом шаге в строку состояния должна

выводиться подсказка о дальнейших действиях пользователя. Программный модуль должен содержать процедуры, выполняющие необходимые действия, обусловленные вариантом задания, и процедуры вывода данных в файл. При старте проекта ввести необходимые данные и провести их обработку. Предусмотреть обработку ошибок. Исходные данные и результаты моделирования сохранить в файле на диске и занести в отчет.

2. Промоделировать физические закономерности, согласно варианту задания. При необходимости преобразовать формулы. Используя приведенные (полученные) соотношения, создать процедуру обработки данных, ввести необходимые значения параметров (если задан диапазон значений параметра, то либо он является независимым аргументом при моделировании, либо его значение необходимо выбирать из указанного диапазона с помощью элементов управления Slider, UpDown) и рассчитать соответствующие зависимости. Вычисления проводить в системе единиц Си.

3. По результатам расчетов построить графики полученных зависимостей (использовать форму и модуль, разработанные при выполнении лабораторной работы Лаб2_11_VB_Plot). Значения параметров и таблицы рассчитанных данных сохранить в файле. В той же папке сохранить график в формате *.jpg.

4.2.3 Варианты заданий

Вариант № 1. Температурная зависимость электропроводности и коэффициента Холла в сильных магнитных полях для примесных полупроводников (пренебречь температурной зависимостью подвижности). Диапазоны изменения параметров: температура полупроводника – $T \in$ [200 -:- 400] K; ширина запрещенной зоны полупроводника – $E_g \in$ [0.7 -:- 2.4] эВ; энергии ионизации доноров и акцепторов – E_d, $E_a \in$ [0.01 -:- 0.3] эВ; концентрации доноров и акцепторов – N_d, $N_a \in$ [0 -:- 1.4]•10^{21} $м^{-3}$; подвижности электронов и дырок – μ_n, $\mu_p \in$ [0 -:- 1.4]•10^2 $м^2с^{-1}В^{-1}$; эффективные массы электронов и дырок – m_n, $m_p \in$ [0.005 -:- 0.4]•m(масса покоя электрона).

Вариант № 2. Зависимость энергетических коэффициентов пропускания и отражения для непоглощающей пленки на непоглощающей подложке от угла падения неполяризованного света при постоянной длине волны. Параметры: показатели преломления: воздуха – $n_0 = 1$; подложки – $n_s = 1.5$; пленки – $n_f = 2.7$ толщина пленки $d_f = d_f \in$ [100 -:- 500] Å; длина волны $\lambda = 750$ нм.

Вариант № 3. Зависимость энергетического коэффициента пропускания для поглощающей пленки на непоглощающей подложке от угла падения неполяризованного света при постоянной длине волны. Показа-

тели преломления: воздуха – $n_0 = 1$; подложки – $n_s = 1.5$; пленки – $N_f = 2.4 – i0.7$. Параметры: толщина пленки $d_f \in [100 -:- 500]$ Å; длина волны $\lambda = 750$ нм.

Вариант № 4. Зависимость энергетического коэффициента отражения для поглощающей пленки на непоглощающей подложке от угла падения неполяризованного света при постоянной длине волны. Показатели преломления: воздуха – $n_0 = 1$; подложки – $n_s = 1.5$; пленки – $N_f = 2.4 - 0.7$ Параметры: толщина пленки $d_f \in [100 -:- 500]$ Å; длина волны $\lambda = 750$ нм.

Вариант № 5. Спектральная зависимость коэффициентов пропускания и отражения для непоглощающей пленки на непоглощающей подложке для неполяризованного света. Показатели преломления: воздуха – $n_0 = 1$; подложки – $n_s = 1.5$; пленки – $n_f = 8 \cdot 10^{-6}\lambda^2 – 0,0109\ \lambda + 6,2288$ Параметры: угол падения $\theta \in [0 -:- 60]°$; толщина пленки $d_f = 5000$ Å;

Вариант № 6. Спектральная зависимость коэффициента пропускания для поглощающей пленки на непоглощающей подложке для неполяризованного света. Показатели преломления: воздуха – $n_0 = 1$; подложки – $n_s = 1.5$; пленки – $N_f = n_f – i\chi_f$; $n_f = 8 \cdot 10^{-6}\lambda^2 – 0,0109\ \lambda + 6,2288$; $\chi_f = 4 \cdot 10^{-6}\lambda^2 + 0,0109\ \lambda + 2,4288$ Параметры: угол падения $\theta \in [0 -:- 60]°$; толщина пленки $d_f = 5000$ Å; длина волны $\lambda \in [200 -:- 1100]$ нм.

Вариант № 7. Спектральная зависимость коэффициента отражения для поглощающей пленки на непоглощающей подложке для неполяризованного света. Показатели преломления: воздуха – $n_0 = 1$; подложки – $n_s = 1.5$; пленки – $N_f = n_f – i\chi_f$; $n_f = 8 \cdot 10^{-6}\lambda^2 – 0,0109\ \lambda + 6,2288$; $\chi_f = 4 \cdot 10^{-6}\lambda^2 + 0,0109\ \lambda + 2,4288$ Параметры: угол падения $\theta \in [0 -:- 60]°$; толщина пленки $d_f = 5000$ Å; длина волны $\lambda \in [200 -:- 1100]$ нм.

Вариант № 8. Темновая вольтамперная характеристика ФЭП по однодиодной модели. Параметры: ток насыщения – $I_s = 10^{-7}$ А; последовательное сопротивление – $R_s = 200$ Ом; шунтирующее сопротивление – $R_{sh} = 5 \cdot 10^4$; фактор идеальности диода – $n \in [2.0 -:- 3.8]$; температура – диода $T \in [280 -:- 380]$ К, прикладываемое напряжение $V \in [-10 -:- 0.8]$ В.

Вариант № 9. Темновая вольтамперная характеристика ФЭП по двухдиодной модели. Параметры: токи насыщения – $I_{s1} = 10\text{-}6$ А, $I_{s2} = 10^{-10}$ А; последовательное сопротивление – $R_s = 200$ Ом; шунтирующее сопротивление – $R_{sh} = 5 \cdot 10^4$; факторы идеальности диодов – $n_1 = 2.0$, $n_2 = 3.8$; температура – диода $T \in [280 -:- 380]$ К, прикладываемое напряжение $V \in [-10 -:- 0.8]$ В.

Вариант № 10. Световая вольтамперная характеристика ФЭП по однодиодной модели. Параметры: ток насыщения – $I_s = 10^{-7}$ А; фототок – $I_{ph} = 54$ мА; последовательное сопротивление – $R_s = 200$ Ом; шунтирующее сопротивление – $R_{sh} = 5 \cdot 10^4$; фактор идеальности диода – $n \in [2.0 -:-$

3.8]; температура – диода T ∈ [280 -:- 380] К, прикладываемое напряжение V ∈ [-10 -:- 0.8] В.

Вариант № 11. Световая вольтамперная характеристика ФЭП по двухдиодной модели. Параметры: токи насыщения – $I_{s1} = 10^{-6}$ А, $I_{s2} = 10^{-10}$ А; фототок – I_{ph} = 54 мА; последовательное сопротивление – R_s = 200 Ом; шунтирующее сопротивление – $R_{sh} = 5 \cdot 10^4$; факторы идеальности диодов – n_1 = 2.0, n_2 = 3.8.0; температура – диода T ∈ [280 -:- 380] К, прикладываемое напряжение V ∈ [-10 -:- 0.8] В.

Вариант № 12. Спектральная зависимость показателя поглощения полупроводника с прямыми разрешенными переходами. Параметры: эффективные массы электронов и дырок – m_e, m_h ∈ [0.005 -:- 0.4]•m(масса покоя электрона); показатель преломления полупроводника n ∈ [2.5 -:- 5.4]; ширина запрещенной зоны полупроводника – E_g ∈ [0.5 -:- 2.4] эВ; энергия кванта света hν ∈ [0.75 -:- 1.24]•E_g.

Вариант № 13. Спектральная зависимость показателя поглощения полупроводника с прямыми запрещенными переходами. Параметры: эффективные массы электронов и дырок – m_e, m_h ∈ [0.005 -:- 0.4]•m(масса покоя электрона); показатель преломления полупроводника n ∈ [2.5 -:- 5.4]; ширина запрещенной зоны полупроводника – E_g ∈ [0.5 -:- 2.4] эВ; энергия кванта света hν ∈ [0.75 -:- 1.24]•E_g.

Вариант № 14. Спектральная зависимость показателя поглощения полупроводника с непрямыми переходами. Параметры: температура Дебая – θ ∈ [400 -:- 900] К; показатель преломления полупроводника n ∈ [2.5 -:- 5.4]; ширина запрещенной зоны полупроводника – E_g ∈ [0.5 -:- 2.4] эВ; энергия кванта света hν ∈ [0.75 -:- 1.24]•E_g.

Вариант № 15. Рассчитать спектральный состав излучения Солнца по формуле Планка (Температура фотосферы 5500 – 6200˚С). Вычислить плотность энергии вблизи орбиты Земли. Параметры: спектральный диапазон – λ ∈ [200 -:- 1100] нм.

Вариант № 16. Рассчитать вольт-фарадную характеристику (зависимость барьерной емкости p-n перехода от напряжения U, приложенного к нему) диода. Параметры: постоянная – A ∈ [0.2 -:- 20] $\cdot 10^{10}$; площадь p-n перехода S ∈ [0.2 -:- 20] $\cdot 10^{-8}$ см^2; постоянная – a ∈ [0.2 -:- 20] $\cdot 10^{19}$ см$^{-4}$; толщина p-n перехода при U = 0 – h_0 ∈ [0.6 -:- 3.0] мкм; германия постоянная B = $1.45 \cdot 10^{15}$ см$^{-3}$К$^{-1.5}$, а для кремния B = $3.48 \cdot 10^{15}$ см$^{-3}$К$^{-1.5}$; ширина запрещенной зоны полупроводника базовой области диода – E_g ∈ [0.6 -:- 1.6] эВ.

Вариант № 17. Распределение неравновесных носителей заряда в образце при их инжекции в момент времени t = 0 в точке с координатой

x = 0 имеет вид: $\Delta p(x,t) = Bt^{-1/2}\exp\left(-\frac{t}{\tau_p}\right)\exp\left(-\frac{(x-\mu_p Et)^2}{4D_p t}\right)$. Параметры: константа – B ∈ [2 -:- 11]•10^{26} $м^{-3}с^{1/2}$; время жизни – τ_p ∈ [2 -:- 2000]•10^{-6} с; подвижность μ_p ∈ [0 -:- 1.4]•10^2 $м^2с^{-1}В^{-1}$; коэффициент диффузии неравновесных дырок – D_p = ; E = U/*l* – напряженность дрейфового поля в образце; U ∈ [0.2 -:- 20] В – напряжение, приложенное к образцу длиной *l* ∈ [2 -:- 20] мм; температура полупроводника – T ∈ [200 -:- 400] K. Проследить поведение Δp(x, t) в диапазоне координат x ∈ [2 -:- 20] мм и времени t ∈ [0 -:- 11]•τp.

Вариант № 18. Спектральная зависимость параметров эллипсометрии Ψ и Δ при отражении от однородной и изотропной среды. Параметры Ψ и Δ определяются с помощью соотношения: $tg(\Psi)\exp(i\Delta) = \frac{r_p}{r_s}$, где r_s и r_p определяются соотношением (2), приведенным в файле «Формулы.doc». Для спектральной зависимости комплексного показателя преломления использовать диапазон длин волн и зависимости, приведенные после соотношения (9) в том же файле. При построении зависимостей использовать угол падения как параметр.

Вариант № 19. Спектральная зависимость коэффициента поглощения полупровод-ника с непрямыми переходами в квантующем магнитном поле. Проследить поведение показателя поглощения для энергий квантов ħω ∈ [1,2 -:- 5] эВ в магнитном поле с индукцией B ∈ [0 -:- 200] Тл, если температура Дебая Θ_D ∈ [120 -:- 550] K; относительные эффективные массы носителей в зоне проводимости m_c ∈ [0,001 -:- 0,1] и в валентной зоне m_v ∈ [0,01 -:- 0,2]; ширина запрещенной зоны полупроводника – E_g ∈ [0.5 -:- 2.4] эВ.

Вариант № 20. Изобразить серию двумерных сечений трехмерной зависимости изоэнергетических поверхностей энергии от импульса в зоне проводимости германия и кремния вблизи минимума энергии.

4.2.4 Содержание отчета

В отчете должны быть представлены:

1. Цель работы.
2. Основные сведения о теме работы.
3. Алгоритм разработанного проекта.
4. Перечень форм и элементов управления.
5. Процедуры обработки событий и данных.
6. Исходные данные и результаты их обработки.
7. Выводы.

4.3 Контрольные вопросы

1. Опишите назначение стандарта имен при написании программ.
2. Что такое "венгерская нотация" при присвоении имени?
3. Какие ограничения действуют при задании имен в Visual Basic?
4. Перечислите префиксы типов данных при присвоении имен переменных.
5. Перечислите префиксы области действия при присвоении имен переменных.
6. Перечислите префиксы, определяющий вид передачи аргумента в процедуру или функцию?
7. Перечислите префиксы имен элементов управления на формах.
8. Опишите назначение опции Option Explicit.
9. Опишите назначение операторы вида DefInt?
10. Опишите назначение структурного форматирования кода.

СПИСОК ЛИТЕРАТУРЫ

1. Сайлер Б., Споттс Дж. Использование Visual Basic 6. Классическое издание. - М.: Вильямс, 2007. – 832 с.
2. Сафронов И. Visual Basic в задачах и примерах. - СПб.: БХВ-Петербург, 2008. – 400 с.
3. Эпплман Д. Win32 API и Visual Basic. Для профессионалов. - СПб.: Питер, 2001. – 1120 с.
4. Сергеев В. Visual Basic 6.0. Наиболее полное руководство для профессиональной работы в среде Visual Basic 6.0. – СПб.: БХВ-Петербург, 2004. – 992 с.
5. Balena F. Programming Microsoft Visual Basic 6.0. – USA: Microsoft Press, 1999. – 1312 p.
6. Halvorson M. Microsoft Visual Basic Professional 6.0 Step by Step. – USA: Microsoft Press, 1998. – 672 p.
7. Holzner S. Visual Basic 6 Black Book: The Only Book You'll Need on Visual Basic. – USA: Coriolis Group Books, 1998. – 700 p.
8. Microsoft Visual Basic 6. Шаг за шагом: Практ. пособ. / Пер. с англ. – М.: Изд. ЭКОМ., Изд. 2-е, исправленное, 2003. – 432 с.
9. Райтингер М., Муч Г. Visual Basic 6.0: для пользователя: пер. с нем. - К.: Издательская группа BHV, 1999. - 416 с.

СОДЕРЖАНИЕ

Учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам
«Основы программирования на Visual Basic»
из раздела «Программирование в среде Visual Basic. Интегрированная среда разработки»
дисциплины «Информатика»
для студентов направления подготовки
6.050801 «Микро- и наноэлектроника»

Составители: ШКАЛЕТО Владимир Иванович
ЗАЙЦЕВ Роман Валентинович
КИРИЧЕНКО Михаил Валерьевич

Ответственный за выпуск Д.А. Кудий

Редактор

План 2013 р.

Підписано до друку _____. Формат 60×84 1/16. Папір друк. №2.

Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 2,5.

Обл.-вид. 3,0. Тираж 50 прим. Зам. № _____. Ціна договірна

Видавничий центр НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.
Свідоцтво про державну реєстрацію ДК № 116 від 10.07.2000 р.

Друкарня НТУ “ХПІ”. 61002, Харків, вул. Фрунзе, 21.